extendsim.

ADVANCED RESOURCE MANAGEMENT

TUTORIAL & REFERENCE

License, Software Copyright, Trademark, and Other Information

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

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

Acknowledgments

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

The contents of this document are the result of years of work by software architects, simulation engineers, and technical writers and editors of ExtendSim products.

ANDRITZ Inc • 13560 Morris Road, Suite 1250 • Alpharetta, GA 30004 USA
770.640.2500 • Info.ExtendSim@Andritz.com
www.ExtendSim.com

Table of Contents

ARM
Tutorial & Reference

Introduction

Welcome!

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

☞ This document assumes you are familiar with discrete event modeling and already know how to launch ExtendSim and build a discrete event model. If not, see the Discrete Event Quick Start Guide.

About this document

The purpose of this document is to get model builders familiarized with the ExtendSim Advanced Resource Management (ARM) system, a powerful framework for representing resources and their organization, properties, requirements, and behaviors. It is used to quantify with a high degree of fidelity the impact of resources on system performance.

⚠ ARM is an advanced feature that is a component of discrete event modeling. It is not available for ExtendSim CP.

Who should read this document

The ExtendSim ARM technology is a sophisticated and comprehensive system for modeling and managing resources.

- **Discrete event model builders**. While it is possible to model resources using other ExtendSim methods, ARM is indispensable whenever a high level of fidelity is required or if complex resource requirements or allocation rules are needed for model integrity.

  Even if those situations don't apply to your models, before deciding how to model resources you should at least read this introductory chapter as well as the "Comparison of explicit resource modeling methods" on page 50. (The User Reference has even more information in its Discrete Event: Resources and Shifts chapter.)

- **Current ARM modelers**. If you already use ARM in your models, there are many new features and there's probably information here that you haven't discovered yet. Especially see "What's new in ARM since 9.0" on page 89.

Chapters in this reference

1) Introduction to ARM (this chapter of the document):
2) Basic information: exploring a model that uses ARM; a glossary of terms
3) Tutorial 1: using the Resource Manager to define and create resources and requirements
4) Tutorial 2: using Excel to define resources and resource requirements

5) Tutorial 3: using an item's attribute value to specify the item's resource requirement or a resource property value
6) Reference: new features since ExtendSim 9.0 and a comprehensive reference

Resource management

A *resource* is a stock or supply of some asset that can be drawn on as needed for an organization to function effectively. Resources are both the factors required to accomplish an activity and the means by which process activities and operations are performed. Examples of resources include equipment, personnel, space, energy, time, information technology, and money.

Use in simulation

In the context of simulation modeling, resource management is concerned with identifying and then appropriately, efficiently, and effectively allocating resources when and where they are needed.

Resources often have, as individual entities or as a group, specific characteristics or properties that distinguish one from another. (For example, personnel could be categorized as sales staff, mechanics, or programmers.) During a simulation, item entities will have a requirement that one or more resources be allocated to them, based on the item's needs and the resource's specific properties.

Simulating resources allows the modeler to:

- Analyze resource availability and utilization, including how efficiently resources are being used
- Determine the impact of resource constraints on the system's capacity—what happens if resources will not be available or if there is a delay until they become available?
- Explore how to improve resource utilization without causing overly long waiting lines
- Determine if waiting lines can be reduced without adding more resources

☞ If there is an infinite supply of a resource, there is no point in explicitly modeling it since it will not cause any constraints on the model.

ExtendSim Advanced Resource Management

ExtendSim provides a unique and sophisticated architecture for systematically dealing with multiple types of resources and complex resource requirements in simulation models. The ExtendSim Advanced Resource Management (ARM) feature is a comprehensive system for:

- Creating resources and distinguishing between them
- Allocating resources to items as required by the model
- Releasing resources for reuse at the end of their assignment
- Generating statistical reports

The ARM system provides straightforward methods for defining complex resource requirements for items as well as a flexible set of rules for how resources get allocated to them.

When you should use ARM

One of the main reasons organizations build models is to determine how much work can get done, how long it will take, and how much it will cost. The processes being modeled are often resource-constrained—items can't complete their activities unless specific resources are made

available to them. For those models it is important to determine what the impact on operations will be, given the available resources and how they're configured.

As discussed in "Comparison of explicit resource modeling methods" on page 50, ExtendSim provides several alternatives for modeling resources. However, the ARM method is preferred when some or all of the following are true:

- A high degree of fidelity is desired or required for the model's resources
- The resources need to be organized or differentiated in complex ways—available times, conditions, training, and so forth
- Human capital resources are cross-trained or cross-functional
- There are complex requirements that determine which types and how many resources items need to complete their tasks
- There are complicated rules for how and when resources can get allocated to items
- You need to collect detailed statistics, such as time spent in different states and utilization, for individual resources.

☞ The ARM system can co-exist with the other resource modeling approaches (e.g. items as resources and non-ARM resource pools). Thus ExtendSim models can use a mixture of resource modeling methods.

Framework

The following components comprise the ExtendSim Advanced Resource Management system:

1) An ExtendSim database (**Advanced Resources)** is a repository for the information used to represent the state and behavior of resources in the ARM system and provides the ARM system's architectural framework. ARM uses this database to represent, manage, and track the status and properties of resources, pools, and groups, as well as allocation policies and release rules.
2) The **Resource Manager block** (Item library) is the central component or "brain" of the system. It:

   * Manages and controls all resource allocations and releases
   * Automatically generates and updates the Advanced Resources database
   * Provides a dialog-based interface for creating and managing resources, requirements, policies, and release rules. (For defining resources and requirements, an Excel workbook can be used as an alternative to the Resource Manager block.)
3) *When set to advanced resource mode*:
   * **Queue blocks** (Item library) communicate with the Resource Manager block to trigger the allocation of resources to its items.
   * **Resource Pool Release blocks** (Item library) communicate with the Resource Manager block to trigger the release of resources from the items they've been allocated to.
4) **Other blocks** (listed on page 82) control or report various aspects of resources.

These components will be shown in the next chapter.

☞ As an alternative to the dialog-based interface of the Resource Manager block, external applications such as an **Excel workbook** can be used to define and manage the model's resources and requirements.

ARM features

ARM has many features that makes it a powerful and sophisticated component of simulation models, including:

Feature	Description
Comprehensive	Create and manage resources and distinguish between each one. Use whole or fractional resources. Create complex resource requirements and release rules, set policies, manage data.
Automated	An ExtendSim database is automatically added to the model and populated with data as resources and requirements are created. A traceable resource order is automatically generated whenever an item requests to have its resource requirement satisfied.
Cost-Assessed	Costs can be tracked per time unit or per use on individual resources; total costs are automatically calculated and reported.
Customizable and Extensible	Resource requirements, release rules, and resource properties are customizable and reusable.
Integrated and Connected	ARM is tightly integrated with blocks in the Item library and the ExtendSim database. Resource information, including allocation rules, is retained within the model. Use Excel or another external application to define resources, resource requirements, and custom properties. Import ARM definitions from external applications to the ExtendSim database.
Scalable	Resource requirements are completely scalable since item attributes can be used as references to resource properties and quantities and even to specify which resource requirement the item will use. ARM also creates default resource requirements, allows for custom resource properties, and allows for the auto-generation of resources, pools, groups, properties, and requirements.
Helpful	Statistics and reports are automatically calculated and stored, including total orders serviced, utilization, total idle/busy/down/disabled/off-shift times, resource state transition events, and much more.

☞ For a glossary of ARM terms, see "Definitions" on page 16.

Where to get more information outside of this document

All documentation is installed as eBooks in the Documents/ExtendSim/Documentation/folder.

User Reference

The ExtendSim *User Reference* has several How To chapters as well as a comprehensive reference section for Discrete Event modeling.

Discrete Event modeling

To learn more about discrete event modeling, see:

- The *Discrete Event Quick Start Guide* for general information and a tutorial on building discrete event models

- The *User Reference's Discrete Event section* and its chapters on items, routing, batching, and more

ExtendSim database

To learn more about ExtendSim database capability, see the *ExtendSim Database Tutorial and Reference*.

Technical Reference

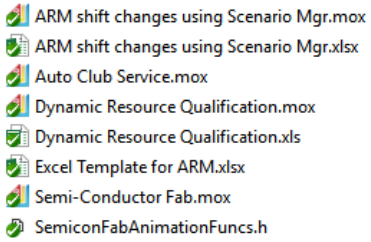
In most cases modelers will use the Resource Manager block or Excel to incorporate ARM in a model. However, the Technical Reference is a great resource when using equation-based blocks or when creating custom blocks for your model. The ExtendSim integrated development environment (IDE) has thousands of pre-defined functions plus provides access to sophisticated tools such as include files, a code editor, conditional compilation, a source code debugger, code completion, and external source code capabilities.

☞ The ExtendSim manuals are eBooks that ship with every ExtendSim product. To access them, see the Documents/ExtendSim/Documentation/folder, select the Help menu when using ExtendSim, or launch the references from the Getting Started "Quick Start" model that opens when ExtendSim launches.

Example models show you how

ExtendSim includes numerous tutorial models as well as example models that explain concepts discussed in the documentation. See the Documents/ExtendSim/Examples folder.

⚠ This document discusses ARM tutorial models. As indicated below, there are several additional ARM models located at Documents/ExtendSim/Examples/Discrete Event/Resources and Shifts/Advanced Resource Mgt. That folder also contains the Excel Template for ARM, which can be used to define resources and resource requirements in Excel.



ARM
Tutorial & Reference

Basics: Exploring ARM

How Advanced Resource Management works

The basic mechanism for creating, allocating, and releasing resources using ARM is:

1) The model has multiple types of resources as well as requirements that specify how many and what types of resources are needed
2) One or more Queue blocks are set to be advanced resource queues
   * Depending on their needs, items in the Queue request resources, then wait until their requirements can be satisfied
   * Once resources are allocated to items, the items are released from the Queue
3) The items retain resources until they are released by Resource Pool Release blocks
4) The management of resource states and quantities and the allocation and release of resources is controlled by a single Resource Manager block
5) As the model runs, discrete transactions occur between blocks specified for advanced resources and the ExtendSim Advanced Resources database.
6) These transactions are driven by item arrivals in the Queue and Resource Pool Release blocks, shift changes, and resource pool quantity changes.

☞ The primary ARM transactions are illustrated and described in more detail starting on page 83

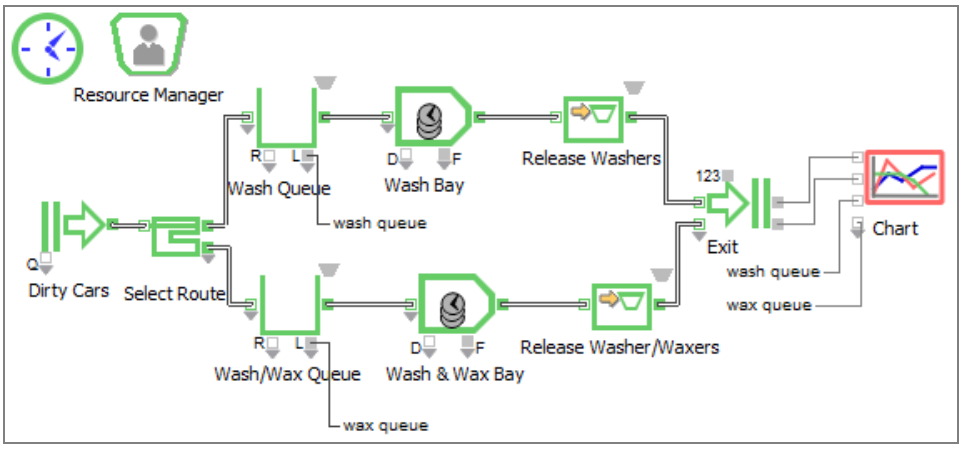
Example model using ARM

To introduce you to the Advanced Resource Management (ARM) methodology and its components, this chapter explores a model that already has ARM. It is the same model you will create in the next chapter.

Open the model

- Launch ExtendSim
- Open the model titled **ARM Tutorial 1 Final** located at Documents/ExtendSim/Examples/Tutorials/Advanced Resources.

☞ Use this model to learn about ARM, but don't save any changes you make because after you build the model in the next chapter, you'll want to compare your model to this original model.



About the model

The example model is derived from the Car Wash models of the tutorials discussed in the *Discrete Event Quick Start Guide*. The car wash has two bays, one for washing and one for washing and waxing. The model simulates cars entering the car wash in one line and then choosing whether they only want washing or also want waxing; 75% of the cars want just washing and 25% want wax and wash.

The main differences between the Car Wash models and the model used here are:

- Blocks specific to implementing ARM—a Resource Manager block and two Resource Pool Release blocks—have been added to the model.
- Each bay has a Queue in front of it, rather than one Queue in the original model, and both Queues have their behavior set to accommodate advanced resources.
- Using ARM, resources (attendants) have already been created for this model: 4 washers in a Wash Only pool and 2 washer/waxers in a Wash & Wax pool.
- Each car requires 2 attendants from a specific pool who will wash (or wash and wax) it as it goes through the bay. Two rules that specify those requirements have already been created.
- Rather than each bay processing one car at a time, an unlimited number of cars can be washed (or washed and waxed) depending on the number and type of available attendants. Thus the capacity of the two Activity blocks (Wash Bay and Wash & Wax Bay) has been changed to infinite.
- Since the model uses ARM, it has an Advanced Resources database that manages all the information.

Exploring the model

Since the model is a simple car wash similar to those described in the Discrete Event Quick Start Guide, the main points to explore in this document are the ExtendSim database and the blocks that are specific to ARM.

ARM Blocks

This model has blocks that are specific to the ARM architecture as well as Queue blocks that are set to use ARM:

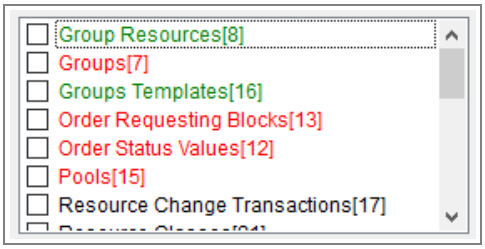
- The Resource Manager block is the central component and user interface for ARM. This block is explored in detail below.
- While not specific to ARM, the two Queue blocks (labeled Wash Queue and Wash/Wax Queue) have their behavior set to *Advanced resource queue*, as seen on page 16. Items will wait in these queues until their required resources get allocated to them.
- The model also has two Resource Pool Release blocks, labeled Release Washers and Release Washer/Waxers. During the simulation run, resources get allocated to items. The only way to release ARM resources for reuse is by passing those items through a Resource Pool Release block. This block is discussed on page 15.

All four blocks come from the Item library.

Advanced Resources database

The Advanced Resources database is an ExtendSim database that stores the information used to represent the state and behavior of resources in the ARM system. (To learn more about the ExtendSim database feature, see the document "ExtendSim Database Tutorial and Reference" located at Documents/ExtendSim/Documentation.)

- In ExtendSim, go to the Database menu and select the Advanced Resources database.
- Scroll through the list of tables and notice that there are tables for pools, resources, resource requirements, and more.



ARM uses this database to represent, manage, and track the status and properties of resources, pools, and groups, as well as allocation policies and release rules. You don't typically make changes directly in this database; the Resource Manager block (discussed below) automatically updates the data in the Advanced Resources database whenever information is entered in the block or the simulation is run.

- Close the database window.

⚠ While it is possible to directly use the database to enable ARM in a model, it is much easier and safer to use the interface of the Resource Manager block.

Resource Manager block

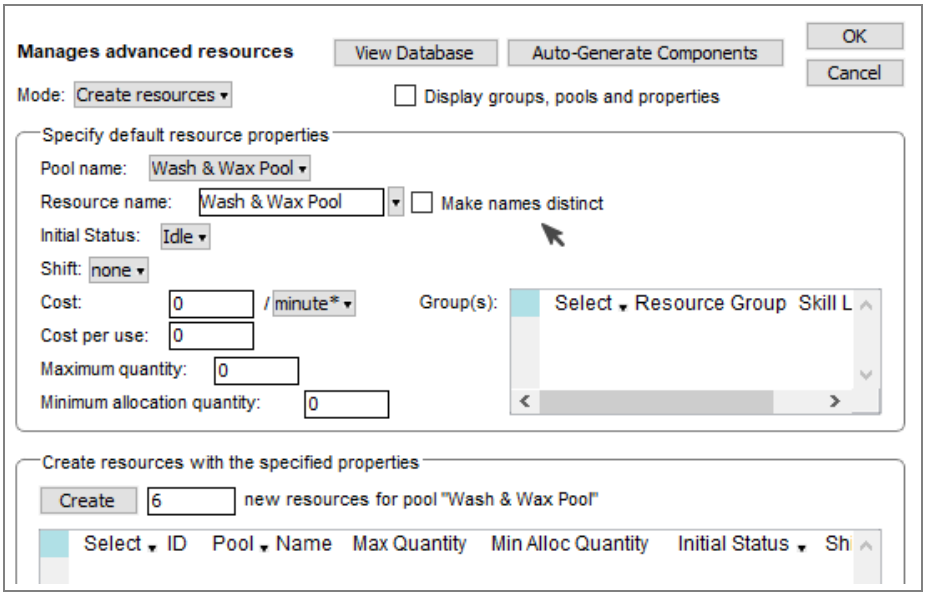
While it is possible to define resources and resource requirements in Excel or some other external application (as will be shown in the next chapter), the Resource Manager block provides the main user interface for ARM.

- Open the dialog of the Resource Manager block.

The block's dialog has tabs for Resources, Requirements, Release Rules, Policies, Event Logging, and Results. Each tab is explained below.

Resources tab

This tab is for creating, editing, and managing resources, pools, and groups.



Resources, pools, and groups

- *Resources* are what items in the model need to proceed to the next step in the process.
- A *pool* is a resource property that is used to organize resources so they can be referred to by a common relation. Since they are used to organize resources, there must always be at least one pool in a model before any resources are created.
- *Groups* are an optional, but powerful, method for organizing resources from one or more pools

☞ These terms are defined on page 16 and further explained in the reference chapter that starts on page 49.

Modes

The Resources tab operates in three modes:

1) *Create resources*. For creating resources, groups, and pools and specifying their properties.
2) *Edit resources*. For editing or deleting resources, groups, and pools, and for changing their properties.
3) *Manage groups*. To define which groups, if any, selected resources belong to. Groups are an optional property of resources and are discussed on page 56.

Resource properties

Resources have *properties*—characteristics which distinguish them from each other and allow them to be completely specified. An example of a resource property is *Pool* or *Initial Status*.

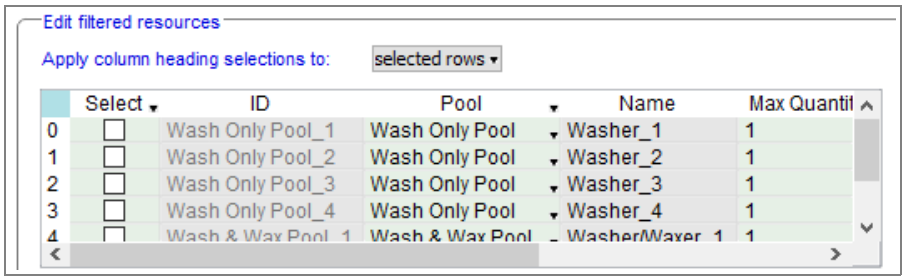
As is true for item attributes, each resource property can have multiple values (for example, the value for Initial Status could be *idle* or *disabled*). Every resource is required to have a Pool

property, a Maximum Quantity, and a minimum Allocation Quantity; the other resource properties (initial status, total down time, etc.) are optional. As discussed in “Properties” on page 61, ARM provides for over 40 predefined resource properties and you can add custom properties.

Explore this model’s ARM resources and pools

▸ In the Resources tab, change to *Mode*: **Edit resources**

The top frame is for filtering resources by selected property values for the pool, initial status, and so forth. With the default settings, all the model’s pools and advanced resources are displayed in the table in the bottom frame, shown here.



As mentioned earlier, this model has two pools and six resources:

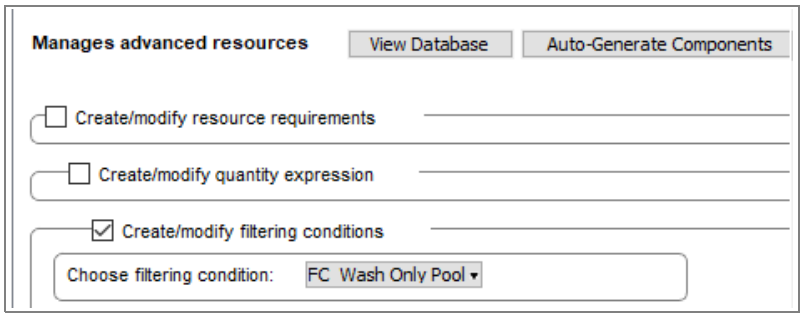
- The Wash Only pool has 4 resources, named Washer_1 to Washer_4
- The Wash & Wax pool has 2 resources, Washer/Waxer_1 and Washer/Waxer_2

In this model all the resources are initially in the Idle status and each of them is represented in the Advanced Resources database as a single, whole resource. (You will learn more about whole and fractional resources in “Fractional and multiple resources” on page 52.)

Requirements tab

▸ Go to the Resource Manager block’s Requirements tab

This tab is for creating named rules that specify how many of what types of resources the items will need. The process of creating a resource requirement involves three steps:



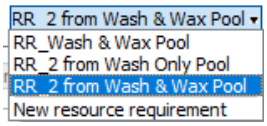
1) Create a filtering condition that limits the universe of all the model’s advanced resources to a specific collection of resources. For example, only those in the Wash Only pool.
2) Create a quantity expression to indicate how many of those specific resources will be needed, such as *2 from the Wash Only pool*.

3) Finally, create the resource requirement. This can be as simple as the specified quantity of resources with a specific condition (2 from the Wash Only pool) or as complex as multiple quantity expressions combined together using AND or OR.

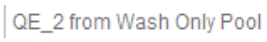
☞ As an alternative to using this tab, resource requirements can be defined in Excel or some other external application and auto-generated by ARM. Both methods are shown in the following chapters.

Explore this model's resource requirements

- In the block's Requirements tab, check the box to **Create/modify resource requirements**
- Click the popup menu for *Choose resource requirement* to see the three resource requirements that have been created for this model:
  - * RR_Wash & Wax Pool
  - * RR_2 from Wash Only Pool
  - * RR_2 from Wash & Wax Pool



- Notice that the expression statement for the selected resource requirement is displayed. For example, selecting *RR_2 from Wash Only Pool* causes the expression *QE_2 from Wash Only Pool* to be shown in the expression field.

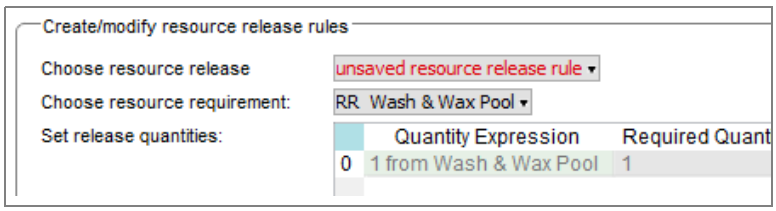


As shown on page 14, based on whichever resource requirement is selected in a Queue block, the ARM system determines which resources, and how many, each item needs before it can be released from the Queue.

Release Rules tab

- Go to the Resource Manager block's Release Rules tab

The Release Rules tab is where custom release rules are created. When used, these rules control which resources are released from the items that pass through the Resource Pool Release blocks.



Why release rules aren't always used

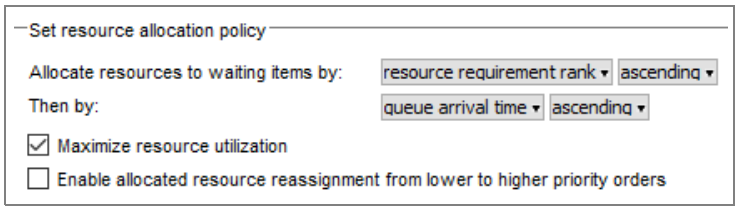
Resources that have been allocated to items can only be released when the items pass through a Resource Pool Release block. Settings in that block provide that it releases either all the resources, the resources from a specific resource order, or specific resources as identified by a custom release rule that was created in the Resource Manager's Release Rules tab. Thus a custom release rule is only used if it has been previously created in the Resource Manager block and the Resource Pool Release block is set to use it.

Custom release rules are optional and this model doesn't use any. For detailed information about release rules, see page 77.

Policies tab

▸ Go to the Resource Manager block's Policies tab

The Policies tab controls the order in which resources search for waiting items. This determines which items a resource can allocate itself to when the resource becomes available for work.



Whenever a resource becomes idle, it has an opportunity to allocate itself to one of the items that are waiting for resources. The settings in this tab are used to sort the list of waiting items before allocation is attempted.

There are several policy options in the top frame. The default is that items that are candidates for allocation are sorted by their resource requirement rank and then by their queue arrival time. If the resource requirement rank is part of the policy, the tab's bottom frame allows you to override the rankings for individual resources.

This model uses the default policy settings. For detailed information about setting policies, see page 78. For information about resource requirement rankings, see page 80.

Event Logging tab

▸ Go to the Resource Manager block's Event Logging tab

If enabled, the Event Logging tab traces specified transaction types while the simulation is running.

As shown on "Logging status transitions" on page 59, if *Resource status transitions* is selected in the Transaction Type table, a Status Transitions table appears to the right with over 100 transitions that can also be traced during the run. For more information, see "Status transitions" on page 58.

Explore this model's event logging

The Order Resources, Allocate Resources, and Release Resources transactions have been selected in this model. To see the results:

- ▸ Click the View Database button in the Event Logging tab
- ▸ Choose Data View in the database window
- ▸ In the All Tables pane, double-click the **Resource Transaction Log** table to open the database Viewer, shown here.

Time[1]	Transaction ype[2]	Order ID[3]	Resource R irement[4]	Result[5]
0.000	Order res...	1	RR__2 fro...	Succeede
0.000	Allocate r...	1	RR__2 fro...	Succeede
6.000	Release r...	1	RR__2 fro...	Succeede
8.513	Order res...	2	RR__2 fro...	Succeede
8.513	Allocate r...	2	RR__2 fro...	Succeede
9.667	Order res...	3	RR__2 fro...	Succeede
9.667	Allocate r...	3	RR 2 fro...	Succeede

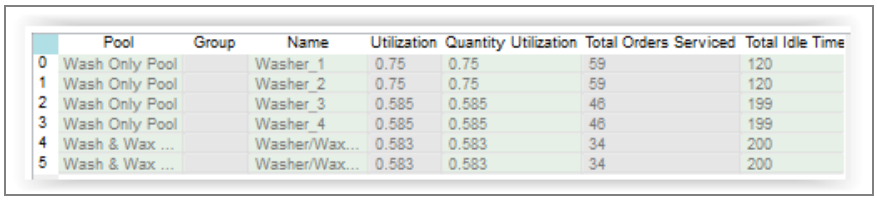
Transaction types are discussed on page 60 and status types and transitions are discussed starting on page 58.

☞ Use logging judiciously. Choosing all the transaction types, and especially using resource names in the logging table, can significantly slow simulations.

Results tab

▸ Go to the Resource Manager block's Results tab

The Results tab provides statistical information about selected ARM components during the simulation run. Choose that this tab reports on all pools or a specific pool, all resources or a specific resource, and so forth. As seen below, the block gathers a lot of information, all of which is also transmitted to the Advanced Resources database.



	Pool	Group	Name	Utilization	Quantity Utilization	Total Orders Serviced	Total Idle Time
0	Wash Only Pool		Washer_1	0.75	0.75	59	120
1	Wash Only Pool		Washer_2	0.75	0.75	59	120
2	Wash Only Pool		Washer_3	0.585	0.585	46	199
3	Wash Only Pool		Washer_4	0.585	0.585	46	199
4	Wash & Wax ...		Washer/Wax...	0.583	0.583	34	200
5	Wash & Wax ...		Washer/Wax...	0.583	0.583	34	200

For this model, the table reports how much each resource was utilized during the run, the total number of resource orders that were serviced, idle time, and so forth.

▸ When you've finished examining the Resource Manager block, close its dialog

Queue blocks

When an item arrives in a Queue that is in ARM mode, the Queue notifies the Resource Manager block that there is an item that needs its resource requirement satisfied. The Resource Manager executes the specified resource requirement—determining which resources, and how many, each waiting item needs and whether those resources are available—and notifies the Queue as soon as the requirement can be satisfied. (As will be shown in Chapter 3, if any part of the resource requirement is implicitly defined—that is, referenced by the item's attribute—the Resource Manager also looks up the item's attribute values.)

For this model, the resource requirements were explicitly defined in the Resource Manager's Requirements tab and each Queue block has been set to use one of those specific requirements. Until the required resources are allocated to an item, it must wait in its Queue. In this manner the ARM system manages the allocation of scarce resources throughout the model.

☞ Not every queue in the model needs to be set to advanced resource behavior, only those whose items require ARM resources.

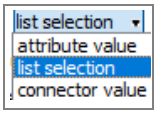
Explore a Queue block

▸ Open the dialog of one of the Queue blocks (Wash Queue or Wash/Wax Queue)

▸ Notice that the behavior has been set to *advanced resource queue*

Items in these queues can't leave the block until their resource requirements are satisfied. The queues order resources based on the selected resource requirement.

The block's dialog gives options for selecting a resource requirement. The requirement can be specified by:



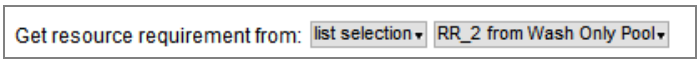
- Picking from the list of the model's existing resource requirements (as shown below and in Tutorials 1 and 2)
- Or, an attribute value (as shown in Tutorial 3)

- Or, a connector value

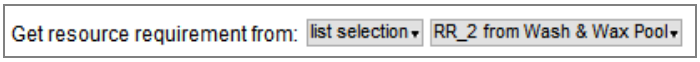
Explore this model's resource ordering behavior

The Queues in this model order resources for their waiting items based on a resource requirement that is selected from a list.

- The Wash Queue is defined such that every item waits for 2 resources (attendants) from the Wash Only pool.



- The Wash/Wax Queue has its items wait for 2 attendants from the Wash & Wax pool.

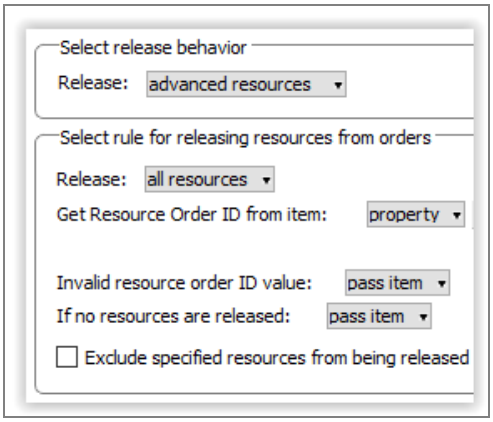


▸ When you're finished examining the Queue block, close its dialog

Resource Pool Release block

As mentioned earlier, allocated resources are only released when the items pass through a Resource Pool Release block.

When an item with ARM resources enters the Resource Pool Release block, the block notifies the Resource Manager that resources need to be released from the item. The Resource Manager releases the resources using release rules specified in the Resource Pool Release block and the item passes through the block, minus the specified resources.



Explore a Resource Pool Release block

▸ Open the dialog of one of the Resource Pool Release blocks (Release Washers or Release Washer/Waxers)

▸ Notice that the behavior has been set to *Release: advanced resources*

Depending on the selection chosen, this block can release, from the items that pass through it:

- All the resources.
- Or, the resources from a specific resource order.
- Or, resources using a custom rule. (Custom release rules are created on the Resource Manager block's Release Rules tab.)

Explore this model's release rules

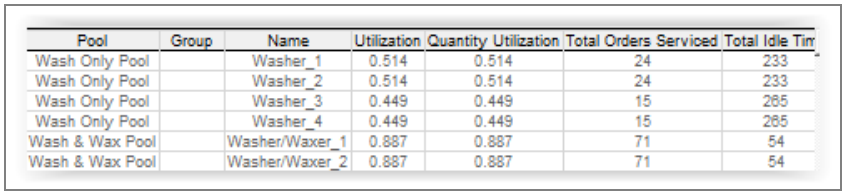
In this model, both of the Resource Pool Release blocks are set such that all the resources will be released from the items that pass through. Released resources transition to the Idle state and become available for assignment to other items in the model.

▸ When you're finished examining the Resource Pool Release block, close its dialog

Running the model

- Run the model
- Go to the Resource Manager's Results tab

As seen below, the top table in this tab captures statistics about each resource.



Pool	Group	Name	Utilization	Quantity Utilization	Total Orders Serviced	Total Idle Tim
Wash Only Pool		Washer_1	0.514	0.514	24	233
Wash Only Pool		Washer_2	0.514	0.514	24	233
Wash Only Pool		Washer_3	0.449	0.449	15	265
Wash Only Pool		Washer_4	0.449	0.449	15	265
Wash & Wax Pool		Washer/Waxer_1	0.887	0.887	71	54
Wash & Wax Pool		Washer/Waxer_2	0.887	0.887	71	54

Definitions

Before going through the tutorials in the following chapters, it is helpful to understand some terms. The following definitions and abbreviations are used regarding the ARM system. Note that italicized words are defined somewhere else in the table.

Term	Definition
Advanced resource (AR)	A *resource* that is designated as part of the Advanced Resource Management (ARM) system.
Filtering condition (FC)	One specified condition of a *property* for a *resource* or a *group*. For example, that the resource comes from a particular pool or that it has a defined cost per use. When applied during the simulation run, filtering conditions restrict the allocatable resources to those that meet the specific conditions. Filtering conditions are saved so they can be used in a *quantity expressions*. They are thus the smallest (atomic) components of a *resource requirement*.
Group	Groups are a secondary and optional method of organizing resources from one or more *pools*. (While a resource can only belong to one pool, it can be a member of none, one, or multiple groups.) Like pools, groups can be used to control which types of resources are allocated to and released from items.
Pool	A resource *property* that is the required method for organizing resources. Each resource must have a pool property to control which types of resources are allocated to and released from items, as well as to capture statistics. A resource can belong to only one pool but it can be a member of many *groups*.
Properties	A method for distinguishing one *resource*, *group*, *pool*, or item from the others. There are over 40 pre-defined resource properties and you can create custom resource properties. A *Pool* is a required property for each resource.
Quantity expression (QE)	A component of a *resource requirement* that specifies the desired number of resources to be selected from a list of those that meet the *filtering conditions*. Each resource requirement has at least one quantity expression that consists of a specified quantity of one or more *filtering condition* results.

Term	Definition
Resource	An economic or productive factor; the means by which process activities and operations are performed. Typical resources include equipment, personnel, space, energy, time, and money.
Resource order	A request by an item to satisfy a *resource requirement*. This causes the Resource Manager to assign the item a unique *Resource Order ID* which is stored in the item property "_AR Order ID".
Resource Order ID	The location (record index) of a resource order in the Resource Orders database table. It is optionally attached to items using attributes in queues.
Resource requirement (RR)	A named rule that specifies how many of what types of *resources* to allocate to an item. The rule could be simple (1 washer/waxer is required per car) or it could be a complex expression involving several resources, groups, and so forth. A resource requirement can be composed using one or more *quantity expressions*.
Resource release rule	A customizable mechanism for controlling which *resources* to release when items enter Resource Pool Release blocks. Each resource release rule is associated with a specific *resource requirement* and is designed to control the release of resources at the *quantity expression* level of the *resource requirement*.

ARM
Tutorial & Reference

Tutorial 1: Resource Creation Part 1

Overview

This chapter shows how to use the Resource Manager block's dialog to define and create resources and resource requirements for a model.

Chapter 2, which starts on page 31, will show how to instead use an Excel workbook to define those resources and resource requirements.

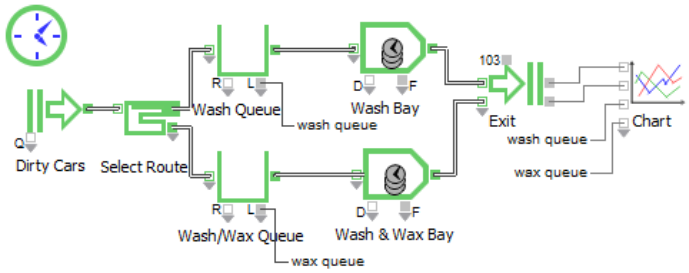
⚠ It is suggested that you go through the Part I tutorial to understand the concepts and the Part II tutorial to learn a different method.

A sample model

The model you will have created by the end of this chapter is the one shown in the previous chapter.

Open the model

- Launch ExtendSim
- Open the model **ARM Tutorial 1** (*not* the Final version) from the folder Documents/ExtendSim/Examples/Tutorials/Advanced Resources
- Save the model as **My ARM Tutorial 1**



⚠ The models used in the ARM tutorials are not scalable and do not use the best modeling techniques. However, they are purposefully structured to make it easier for you to learn ARM concepts. For a scalable, logical example see the *Semi-Conductor Fab* model located at ExtendSim/Examples/Discrete Event/Resources and Shifts/Advanced Resource Mgt. It shows how to use ARM with 2D animation of a material handling system.

Tutorial assumptions

This model is based on the Car Wash models discussed in the *Discrete Event Quick Start Guide*. However, it assumes that attendants are needed to wash (or wash and wax) the cars as they proceed through the bays. To add ARM to this model, you need the following information:

- There are two types of attendants:
  - Washers can only work on cars that need washing but not waxing
  - Washer/Waxers can only work on cars that need waxing as well as washing
- Each type of attendant will be stored in a separate ARM pool
- There are 4 Washers and 2 Washer/Waxers
- Each car requires either 2 Washers or 2 Washer/Waxers, as appropriate
- Each bay can accommodate any number of cars

Since attendants are restricted to a specific bay, each type of attendant will have its own pool. If instead the attendants could move between bays (e.g. if they were cross-trained), the model could have one pool for all the attendants and divide them into groups (discussed on page 56). But lets keep this model simple.

Add a Resource Manager block

▸ Place a Resource Manager block (Item library) in your model. The block can be placed anywhere, but the top left of the model will provide easiest access.

As you can see from the Database menu, adding a Resource Manager block automatically adds a pre-configured Advanced Resource database to the model. This database has tables, fields, and records to hold all the information that is created when you add ARM to a model and when you run the simulation.

⚠ The Resource Manager is only available in the ExtendSim DE and ExtendSim Pro products.

Create and configure the resources

Resources can have one or more properties, but a pool must be one of those properties. So the first step is to create a pool to hold the Washer resources.

Open the Resource Manager block

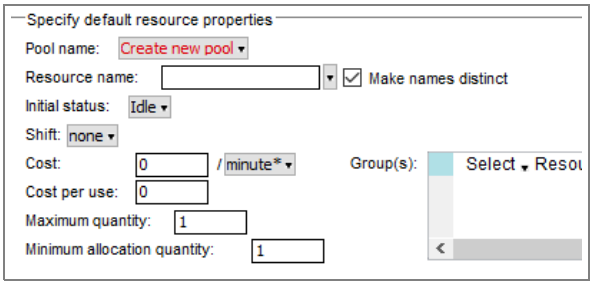
▸ Open the Resource Manager block's dialog

▸ Label the block "Resource Manager"

☞ Other than the Resources tab, the other tabs in this block are disabled. The tabs are meant to be used sequentially and will remain disabled until you've created some advanced resources.

Create a pool of attendants who only wash cars

By default the dialog of the Resource Manager block is set to *Mode: Create resources*, which is what you want.



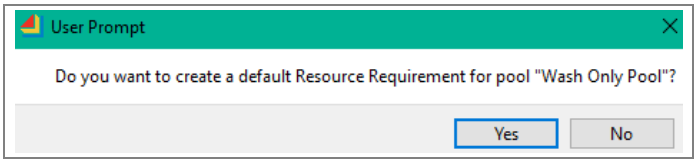
The top frame is where you specify default resource properties—the pool and so forth—for the resources you will create.

Create a pool

Since this model doesn't have any resource pools, and since each resource is required to have a pool as one of its properties, the first step is to create a pool property for the attendants.

Pools can be created in the Resource tab's top frame (*Specify default resource properties*) or by checking the box to "Display groups, pools, and properties".

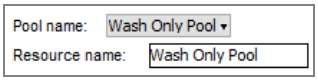
- In the Resource's tab, go to the top frame:
  - Click the popup menu for *Pool name:* **Create new pool**.
  - In the dialog that appears, name the pool **Wash Only Pool** and click OK
  - In the message that appears, choose **No**, so that a default resource requirement is not created by default. (For this part of the tutorial, you will create it manually so you can see how it's done.)



Name the resources

Whichever name is entered in the *Resource name* field will be the root name for all the resources created for this pool. In addition, whichever properties are set here will apply by default to every resource in the pool. (Resource names and other properties can be changed later.)

- As shown here, by default the resource name (Wash Only Pool) is taken from the name of the pool. Click in the field and change the resource name to **Washer**.



Specify the other default properties

The properties in this first frame will apply by default to all the resources in the pool. They can be changed later for individual resources.

- Leave the resource properties (initial status, shift, cost, and so forth) at their defaults.

What happened

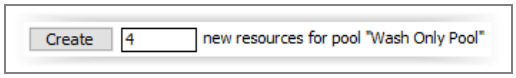
The resources have been defined as:

- Having a Pool property named Wash Only Pool
- Having the Idle status with no Shift or cost information

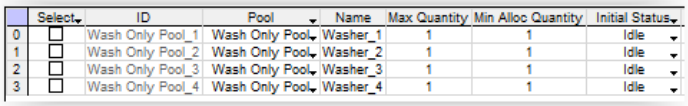
However, the ARM system doesn't yet know how many resources will have that Pool property and if they will all have the same default properties or if some of them will have different default properties.

Create resources for the Wash Only pool

- In the bottom frame (*Create resources with the specified properties*):
  - Enter the number **4**
  - Then click the **Create** button



As shown below, this creates 4 resources (Washer_1 through Washer_4) that each have a Pool property named "Wash Only Pool" and the default initial status and other properties.

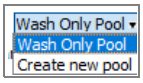


	Select	ID	Pool	Name	Max Quantity	Min Alloc Quantity	Initial Status
0	☐	Wash Only Pool_1	Wash Only Pool	Washer_1	1	1	Idle
1	☐	Wash Only Pool_2	Wash Only Pool	Washer_2	1	1	Idle
2	☐	Wash Only Pool_3	Wash Only Pool	Washer_3	1	1	Idle
3	☐	Wash Only Pool_4	Wash Only Pool	Washer_4	1	1	Idle

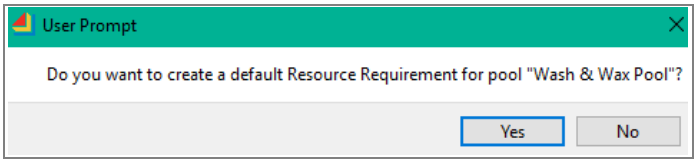
- Save your model

☞ The resources and their properties are automatically placed in the Advanced Resources database. Saving the model saves those database changes.

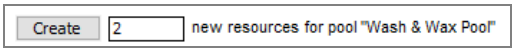
Create a second pool of attendants who only wash and wax cars



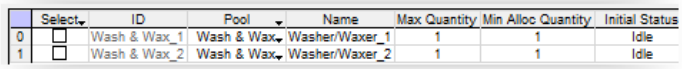
- Click the *Pool name* popup menu again. Since you've already created one pool, the *Pool name* popup menu shows the existing pool.
- Select **Create new pool**
- Name the new pool **Wash & Wax Pool** and click **OK**
- Choose **Yes** when asked if you want to create a default Resource Requirement for that pool



- Change the resource name to **Washer/Waxer** but leave the other options in the top frame at their defaults
- In the frame for creating resources, enter the number **2** and click the **Create** button



This creates two Washer/Waxer resources, each of which has the Wash & Wax pool and the Idle status as its properties.



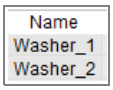
	Select	ID	Pool	Name	Max Quantity	Min Alloc Quantity	Initial Status
0	☐	Wash & Wax_1	Wash & Wax	Washer/Waxer_1	1	1	Idle
1	☐	Wash & Wax_2	Wash & Wax	Washer/Waxer_2	1	1	Idle

Save the model

- Click OK to close the dialog of the Resource Manager block.
- Save the model to save your changes. This also saves the changes to the Advanced Resources database, which now contains all the information you've entered about the pools and their resources.

What happened

As the resources were created, the ARM system automatically:



- Named the resources (for example Washer_1 and Washer_2)
- Assigned the specified properties to them (such as the Wash Only Pool)
- Stored the resources as records in the Resources table of the Advanced Resources database
- Listed them in the Resource Manager block so they can be edited

To view all the resources you've created, along with their properties, change the Resource Manager block's Resources tab to *Mode: Edit Resources.*

☞ The resources you've created here have only one property that you've explicitly specified—their pool. It is more common that a resource would have multiple, compounded properties (e.g. a particular pool plus a specific skill plus a utilization percentage of less than a certain amount).

Set requirements for washers

The next step is to create a resource requirement which will establish how many of what types of resources to allocate to an item. In this case, two attendants who come from the pool Wash Only.

Resource requirements are built in three steps.

1) The first step is to create and save one or more *filtering conditions*. Each filtering condition is one specific property that a resource must have, or not have, in order to be selected for a requirement that references the filtering condition. For example, a filtering condition could be that the resource comes from a particular pool or that it doesn't cost more per use than some defined amount. Filtering conditions are explored in detail starting on page 69.

2) The second step is to define and save one or more *quantity expressions*. Each quantity expression is a logical statement—the desired number of resources that will be required for the task, where the collection of resources is limited by the use of one or more filtering conditions. An example is "2 workers from pool A". Quantity expressions are discussed starting on page 71.

3) The goal is to arrive at a *resource requirement*. In the simplest case, the resource requirement could be the same as the quantity expression (e.g. "2 workers from pool A"). It could also be a complex expression that uses multiple quantity expressions to select resources with specific properties from several pools and/or groups. For example, "2 workers from pool A and 1 worker who costs less than $100 per use". See page 67 for more information.

Create a filtering condition for the Washers

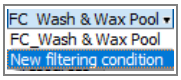
The filtering condition in this tutorial will be that the resources must come from a specific pool—Washers from the Wash Only Pool, Washer/Waxers from the Wash & Wax Pool.

A new filtering condition

- Go to the **Requirements tab** of the Resource Manager block
- Check the box to *Create/Modify filtering conditions*

☞ Notice that the popup in the *Choose filtering condition* frame already lists one filtering condition—FC Wash & Wax Pool. This is because you chose on page 22 to allow the system to create a default resource requirement for the Wash & Wax pool. You will use this filtering condition later.

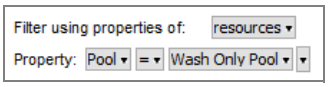
- In the *Choose filtering condition* frame:
  - Click the popup menu
  - Choose **New filtering condition**



In the block's dialog, the popup changes to *unsaved filtering condition* and the text is in red – a reminder that the filtering condition must be saved to be usable.

Filter using the value of a Pool property

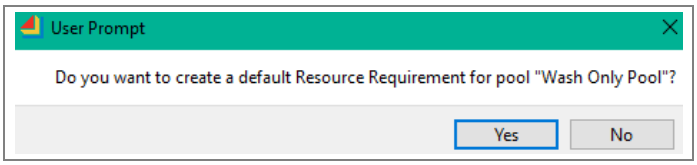
- Choose to **Filter using properties of: resources** (the default)
- Change the properties popup menu to: **Property: Pool**, telling ARM you want to filter using one of the Pool property's values
- Leave the operator as **"="**
- In the popup menu, choose the **Wash Only Pool** as the value for the Pool property. The expression should now read as shown to the right.



☞ Each property can have one or more values. For example, you've created two values for the Pool property: *Wash Only Pool* and *Wash & Wax Pool*.

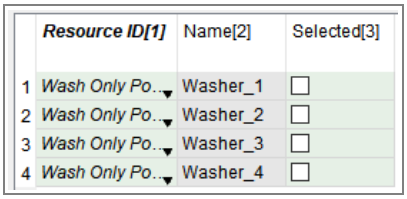
Save

- Click **Save As**, save the filtering condition as **FC_Wash Only Pool**, and click OK.
- Since you want to understand the process of how a resource requirement is created, in the message that appears, choose **No**. This means that a default resource requirement is not created when the filtering condition is created.



Test

- Click **Test**. This opens the ExtendSim database Viewer, displaying which resources are currently in the particular pool that was chosen to filter the resources. (Resources can be added or deleted later.)
- Close the database Viewer window
- Save the model to save your work



	Resource ID[1]	Name[2]	Selected[3]
1	Wash Only Po..	Washer_1	☐
2	Wash Only Po..	Washer_2	☐
3	Wash Only Po..	Washer_3	☐
4	Wash Only Po..	Washer_4	☐

Create a quantity expression for Washers

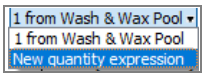
The quantity expression section is where you specify how many resources, from a particular collection of filtered resources, will be needed to complete the task. The filtering condition you just created will be used to narrow the collection of resources to just the ones in the Wash Only pool.

New quantity expression

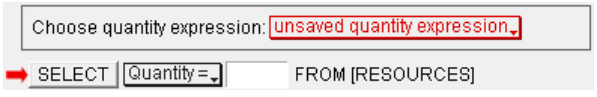
- Check the box to enable **Create/modify quantity expression**

☞ There is already one quantity expression—1 From Wash & Wax Pool—because earlier you said yes to the option to create a default resource requirement for the Wash & Wax Pool.

- In the section *Choose quantity expression*, click the popup menu
- Choose **New quantity expression**



The expression popup changes to "unsaved quantity expression" and the text is in red – a reminder that it must be saved to be usable.



Enter a quantity

- Enter **Quantity = 2**
- Click the **SELECT** button (on the left)

This places the expression in the expression area, indicating that 2 resources will be selected. Since there is no limit on the *type* of resources, any two resources from the Resources database table could be chosen.

```
SELECT Quantity =2 FROM [RESOURCES]
```

Create an expression

To limit the resources to those specified by a filtering condition:

- Click **WHERE**, causing it to be placed in the expression area
- From the *Insert resource filtering condition* popup, choose **FC_Wash Only Pool**

The expression now indicates that two resources will be selected from the pool named *Wash Only*.

```
SELECT Quantity =2 FROM [RESOURCES] WHERE [RESOURCES].[Pool] = "Wash Only Pool"
```

Save and have the system generate the resource requirement

- Click **Save As**
- Save the quantity expression as **QE_2 from Wash Only Pool**

A dialog appears asking if you want a resource requirement to be created from this quantity expression:

> Quantity expression "QE_2 from Wash Only Pool" has been saved. Click Yes if you would like to automatically create a resource requirement based on this quantity expression.

1) *Yes* means you want the resource requirement to be the same as what is described in the quantity expression—any two resources as long as they are from the pool named Wash Only, which is what you want.
2) *No* means you want a resource requirement that is more complex than what is described in the quantity expression.

(For more information about default resource requirements, see page 76.)

- In the dialog, choose **Yes** and change the resource requirement name to **RR_2 from Wash Only Pool**. Then click OK.

Test

- Click **Test**. This opens the database Viewer, displaying which resources would probably be the first selected for allocation. As indicated in the table, without any other conditions being present the system will choose the first two resources in the list.

	Resource ID[1]	Name[2]	Selected[3]
1	*Wash Only Pool_1*	Washer_1	☑
2	*Wash Only Pool_2*	Washer_2	☑
3	*Wash Only Pool_3*	Washer_3	☐
4	*Wash Only Pool_4*	Washer_4	☐

- Close the database Viewer
- If you want more space to work, close the *Create/modify quantity expression* frame by unchecking the check box

Create a resource requirement for Washers

Since the resource requirement was created (above) from the quantity expression, there is nothing more you need to do. In real life however, the section for creating and modifying resource requirements is where you would combine quantity expressions to create complex requirements.

- Click OK to close the dialog of the Resource Manager block and save its changes
- Save the model

Set requirements for Washer/Waxers

When you created the Wash & Wax Pool (see page 22), you also created a default resource requirement of *1 Washer/Waxer from the pool named Wash & Wax*. Using the ARM system to create default resource requirements saves a lot of time if the required quantity is 1. But what happens if, as in this case, the required quantity is 2 attendants?

In the Quantity Expressions section of the Resource Requirements tab, the number of resources included in a quantity expression can be changed to any of the options given by the Quantity button: all filtered resources, quantity from an attribute, or quantity from user input.

☞ This means you could easily change the quantity from a static number to a number indicated by an item's attribute. However, that is beyond the scope of this particular tutorial.

Change the quantity to 2 Washer/Waxers

- Go to the Resource Manager's *Requirements* tab
- Check the box to **Create/modify quantity expression**
- In the quantity expression popup menu, choose the quantity expression the quantity expression that was automatically generated: **1 from Wash & Wax Pool**
- For the Quantity, enter the number **2**

☞ The quantity in the expression will not change until you choose either Save As or Save, as discussed below.

- Assuming you don't want the original expression (1 from Wash & Wax Pool) to now require 2 resources but not change its name, click the **Save As** button. This changes the quantity to 2 resources for the quantity expression, as shown below.
- Name the new expression **QE_2 from Wash & Wax Pool**.

- When asked if you want to generate a resource requirement from that quantity expression, click **Yes**
- Name the new resource requirement **RR_2 from Wash & Wax Pool** and click OK
- Save the model

The above process allows you to change the quantity of an automatically generated quantity expression and create a new resource requirement without having to redo everything.

☞ To change the quantity without changing the name, just click the Save button instead of Save As. This writes over the original quantity expression without changing its name.

⚠ The two resource requirements you created above have the required resource properties and quantities explicitly defined. There is a more scalable approach, as discussed in "Part I: using an item's attribute value to select a resource requirement" on page 41.

Track some transactions

As discussed on page 13, the Resource Manager's Event Logging tab traces specified transaction types while the simulation is running.

- In the Resource Manager's Event Logging tab, check the box to **Enable resource transaction logging**
- Select the following transaction types:
  - Order resources
  - Allocate resources
  - Release resources
- Close the dialog of the Resource Manager block

Later, when the simulation is run, the ARM system will track these transactions and post the data to the Resource Transaction Log table in the Advanced Resources database, shown below.

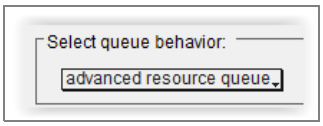
Time[1]	Transaction Type[2]	Order ID[3]	Resource Requiremer	Result[5]	Result Details[6]
0.000	Order res...	1	RR_2 fro...	Succe...	Order successfully processed for item [1] in Queue block [5]
0.000	Allocate r...	1	RR_2 fro...	Succe...	Allocated 2 resources [1,2] with capacities [1,1] to item [1] in Queue...
1.915	Order res...	2	RR_2 fro...	Succe...	Order successfully processed for item [2] in Queue block [5]

Modify Queues to accommodate advanced resources

Not every queue in a model needs to be or should be an Advanced Resource Queue. However, the two queues in front of the bays need to use advanced resources because they will hold cars until attendants are available.

Change the Wash Queue behavior for ARM

- In the dialog of the Wash Queue block, change its queue behavior from a sorted queue to an **advanced resource queue**.



☞ You cannot change a Queue into an advanced resource queue until a Resource Manager block has been placed in the model.

Tell the Wash Queue which resource requirement applies to its items

When set to behave as an advanced resources queue, the Queue releases an item when the item's resource requirements are met. The Queue gets the resource requirement from a list of resource requirements, a connector value, or an attribute value. For this model, you want to get it from the resource requirement named **RR_2 from Wash Only Pool**, which is in the list selection.

- In the section for defining the resource ordering behavior, choose to **Get resource requirement from: list selection** (the default)
- In the popup menu, choose **RR_2 from Wash Only Pool**

- Click OK to close the dialog and save changes

Set the Wash/Wax Queue to use ARM

- Using the same steps as above:
  - Change the Wash/Wax Queue's behavior to be an **advanced resource queue**
  - For the resource requirement, select **RR_2 from Wash & Wax Pool**
- Click OK to close the dialog and save changes
- Save your model

⚠ If resource requirements are explicitly set as done here, every Queue block that is used for ARM would need to have a resource requirement specified in its dialog. See "Part I: using an item's attribute value to select a resource requirement" on page 41 for a more scalable approach.

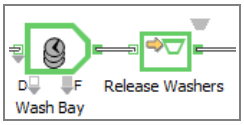
Add Resource Pool Release blocks

If resources get consumed by the items that require them, they don't need to be released for reuse; instead they just pass out of the system. These types of resources are part of an open system, as discussed in the *Discrete Event Quick Start Guide*. Examples of consumable resources are stock or a fixture that gets used up during the manufacturing process.

The attendants for this model are part of a closed system; after they've completed their work they need to be returned to the pool as available resources.

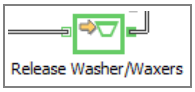
☞ Rather than using multiple Resource Pool Release blocks, one block could release all the resources on the items that pass through it. See Chapter 3 for an example.

Wash Bay



- Insert a Resource Pool Release block (Item library) between the Wash Bay and the Exit block.
- In its dialog:
  - For the release behavior, choose to **Release: advanced resources**
  - Leave the default settings as they are, causing the block to release all the resources from items that exit
  - Label the block **Release Washers**
  - Click OK to close the dialog

Wash and Wax Bay

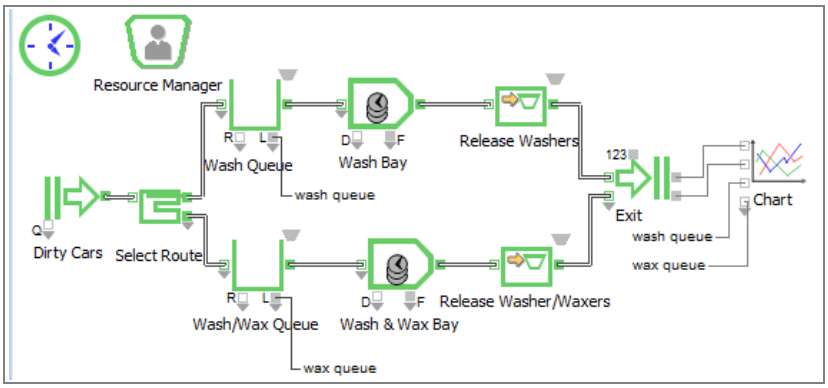


- Insert a Resource Pool Release block (Item library) between the Wash & Wax Bay and the Exit block.
- In its dialog:
  - For the release behavior, choose to **Release: advanced resources**
  - Leave the default settings as they are
  - Label the block **Release Washer/Waxers**
- Click OK to close the dialog

- Save the model

Results

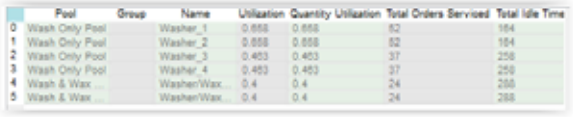
Your model should now look like the one below.



- Run the model

Results tab of the Resource Manager block

- Go to the Results tab of the Resource Manager block



As seen in the table at right, statistics have been calculated for each attendant. Since the attendants work in pairs, it makes sense that each set of pairs would have the same numbers.

A second table in the Results tab gives statistical information for the two pools. Notice that since two attendants are required, each of them will cause the pool's Total Orders to increment by 1. And when washing a car takes 6 minutes and requires two workers, the Total Busy Time will be incremented by 12 minutes for each car washed,

Event Logging tab of the Resource Manager block

The Order Resources, Allocate Resources, and Release Resources transactions have been selected in this model. The details logged for these transactions are helpful when verifying that the proper resources are being allocated to and released from items and at the correct time.

Time[1]	Transaction ype[2]	Order ID[3]	Resource R irement[4]	Result[5]
0.000	Order res...	1	RR__2 fro...	Succeede
0.000	Allocate r...	1	RR__2 fro...	Succeede
6.000	Release r...	1	RR__2 fro...	Succeede
8.513	Order res...	2	RR__2 fro...	Succeede
8.513	Allocate r...	2	RR__2 fro...	Succeede
9.667	Order res...	3	RR__2 fro...	Succeede
9.667	Allocate r...	3	RR__2 fro...	Succeede

To see the results:

- Click the **View Database** button in the Event Logging tab

- In the All Tables pane, double-click the **Resource Transaction Log** table to open the database Viewer, shown here.

☞ The final model is named ARM Tutorial 1 Final and is located at Documents/ExtendSim/Examples/Tutorials/Discrete Event/Advanced Resources.

Next chapter

Instead of using the Resource Manager block to create resources and resource requirements, you can use an external application to define them, then import them to the model.

The next chapter shows how to use an Excel workbook as an interface for defining the same resources and resource requirements you created in this chapter.

ARM
Tutorial & Reference

Tutorial 2: Resource Creation Part 2

Overview

This chapter uses an Excel workbook to define the same resources and resource requirements that you created in the previous chapter. After the definitions are imported to the model, the ARM system uses them to create the model's resources and resource requirements.

Defining resources and resource requirements externally

The previous chapter used the Resource Manager block to *define and create* resources and resource requirements. Resources and resource requirements can also be *defined* in external applications such as Excel, SQL Server, and Access. After the definitions have been imported into the Advanced Resources database, ARM *creates* the resources either the first time the simulation is run or when the resources are auto-generated.

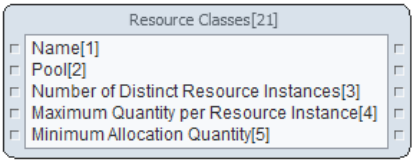
This chapter shows how to define resources and resource requirements using tables in an Excel worksheet, import the definitions to the model, and auto-generate the resources.

⚠ You need Excel to do this tutorial.

Why and how it works

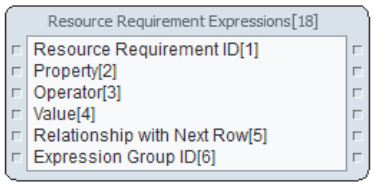
The key is that the ARM database (Advanced Resources) has tables that organize and keep track of everything about advanced resources. Two of the tables in that database are:

- Resource Classes. This database table provides a template for defining all the resources and their associated properties in an ARM model. The records in this table track the name of each resource, the pool it is in, the number of resources of that name in that pool, the number of resources represented by each record (Maximum Quantity per Resource Instance), and the smallest quantity of a resource that can be allocated to a resource order (Maximum Allocation Quantity).



☞ If you add fields to this table, it can also be used to define groups and custom resource properties. Note that each new field must be the name of an included or custom property.

- Resource Requirement Expressions. This table holds the definitions for a model's resource requirements, quantity expressions, and filtering conditions. Each record in the table tracks all or a portion of a resource requirement, including the property used as filtering condition and the quantity used in the quantity expression. The Expression Group ID lets the ARM system know where each expression starts and where it ends.



When a model with ARM is simulated (or the Auto-Generate Components button is clicked), the data from the external application is imported and sent to these two database tables. Every time the simulation is run, ARM checks the Resource Classes and Resource Requirement Expressions tables. If these tables have data in them, ARM auto-builds the resources and requirements from scratch using that data. If the tables are empty, ARM uses whatever is already in the system.

Tutorial assumptions

This exercise assumes that you have created a model that has been partially pre-configured for ARM. So that you can focus on the purpose of this tutorial:

- The model has a Resource Manager block and a Resource Pool Release block
- The two Queues have been configured to use advanced resources
- There are 4 Washer resources

However:

- The model does not have any Washer/Waxer resources or requirements for those resources. You will define those in Excel.
- While a resource requirement for washers has been *defined* in Excel, the requirement has not yet been *created*. You need to import that definition, as well as the definition for the Washer/Waxer resources and the resource requirements into the ARM system.

Working with the Excel workbook

ExtendSim ships with an Excel workbook (*Excel Template for ARM*) for defining resources and resource requirements. This serves as a template you can use as is or modify, for example by adding columns for defining custom properties. The file is located at Documents/ExtendSim/Examples/Discrete Event/Resources and Shifts/Advanced Resources.

For this tutorial you will use a slightly different Excel file that already includes some information for the Washer resources but needs additional information for Washer/Waxer resources and requirements.

- Create a folder in which to save the files for this example, so that the Excel file and the model file are saved in the same folder.
- Launch Excel
- Open the Excel file named **Excel for ARM Tutorial.xlsx**. It is located at Documents/ExtendSim/Examples/Tutorials/Advanced Resources.
- So that you don't overwrite the example, give the Save As command:
  - Name the Excel file **My Excel for ARM Tutorial.xlsx**

- Save it to the folder you created

⚠ The Excel file and the model that uses it must be kept in the same folder location.

Specify the resources

- In Excel, go to the Resource Classes worksheet

Notice that information for the Washer resource has already been entered. The example model also needs a Washer/Waxer resource, so you will enter that in row #3.

	A	B	C	D	E
1	Name	Pool	Number of Distinct Resource	Maximum Quantity per	Minimum Allocation Quantit
2	Washer	Wash Only Pool	4	1	1
3					

- In row #3:
  - Name the resource **Washer/Waxer**
  - Name the pool **Wash & Wax Pool**
  - For the number of instances, enter **2**
  - *Maximum Quantity:* **1**
  - *Minimum Allocation:* **1**
- Save the file

This names the Washer/Waxer resource and its pool property. It also specifies how many Washer/Waxers reside in that pool and indicates that those resources are neither fractional nor multiple. If you've entered the information correctly, the Resource Classes worksheet should look like what is shown here.

	A	B	C	D	E
1	Name	Pool	Number of Distinct Resource	Maximum Quantity per	Minimum Allocation Quantit
2	Washer	Wash Only Pool	4	1	1
3	Washer/Waxer	Wash & Wax Pool	2	1	1

Specify the resource requirements

- Go to the Resource Requirement Expression worksheet

Notice that rows #2 and #3 have already been completed. Those rows specify the requirements for the resources that have the Wash Only Pool as a property. The requirements for the Washer/Waxer resources need to be added to rows #4 and #5.

	A	B	C	D	E	F
1	Resourc	Property	Operator	Value	Relation	Expressic
2	RR_2 from '	Quantity	"="	2	AND	EG_1
3	RR_2 from '	Pool	"="	Wash Only Pool		EG_1

Pre-built linked lists

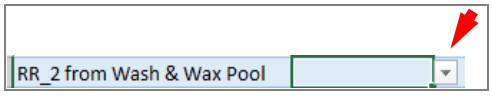
This worksheet has pre-built linked lists with data validation for the following columns:

- Property
- Operator
- Relationship with Next Row

Once you've named a resource requirement, clicking in the cells for Property, Operator, or Relationship will reveal a drop down list for making selections. The drop down lists include all the pre-built options for each of those columns. To add custom options, use the tables in the Lists worksheet.

Row 4

- For row #4:
  - For the Resource Requirement ID, enter **RR_2 from Wash & Wax Pool**
  - Click in the cell for the Property column so that the drop down list selector (the downward arrow) appears to the right of the cell, as shown here



  - Click on the drop down list selector and from the options, choose **Quantity** as the property
  - Similarly, click in the Operator column and choose **"="** from the drop down list
  - Since each car requires 2 attendants, enter **2** for the value
  - From the drop down list for the Relationship column, choose **AND** for the relationship
  - For the Expression Group ID, enter **EG_2**

This defines a part of the expression indicating that 2 Washer/Waxer resources are required; the AND indicates that there is more to the expression coming on the next line.

☞ You can use any wording you want for the Expression Group ID, up to 256 characters. Using the same wording in two or more rows is the equivalent of putting parentheses around the included rows.

Row 5

- For row #5:
  - Copy the Resource Requirement ID from row 4 and paste it in row 5
  - For the property, choose **Pool**
  - Choose**"="** for the operator
  - Enter **Wash & Wax Pool** for the value
  - Since there won't be any further conditions to include in the expression, leave the relationship blank
  - For the Expression Group ID, enter **EG_2**

The result

The result of the information entered in rows 4 and 5 is an expression that is the equivalent of "*RR_2 from Wash & Wax Pool* has as its quantity and conditions 2 resources that come from the Wash and Wax Pool".

If you've entered the information correctly, the Resource Requirement Expression worksheet should look like the worksheet shown here.

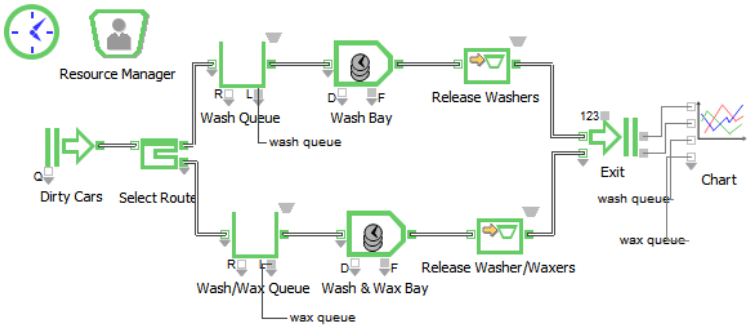
Resource Requirement ID	Property	Oper	Value	Rela	Exp
RR_2 from Wash Only Pool	Quantity	"="	2	AND	EG_1
RR_2 from Wash Only Pool	Pool	"="	Wash Only Pool		EG_1
RR_2 from Wash & Wax Pool	Quantity	"="	2	AND	EG_2
RR_2 from Wash & Wax Pool	Pool	"="	Wash & Wax Pool		EG_2

Save your work

- Save the Excel file

Working with the model

- Launch ExtendSim
- Open the model named **ARM Tutorial 2**. It is located in the folder Documents/ExtendSim/Examples/Tutorials/Advanced Resources



- Save the model as **My ARM Tutorial 2**

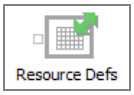
⚠ The model and the Excel file it uses must be kept in the same folder location, so save to the same location you saved *My Excel for ARM Tutorial.xlsx*.

Importing definitions to the database

Data Import Export blocks are used to import the definitions for the resources and resource requirements into the Advanced Resources database.

Import the resource definition to Resource Classes

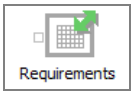
- Add a **Data Import Export block** (Value library) to your model. It can be placed anywhere on the worksheet.
- In the dialog of this Data Import Export block, leave the block's action as the default **Import from Excel spreadsheet** to **ExtendSim database**
- Specify the ExtendSim data target as:
  - *DB:* **Advanced Resources**
  - *Table:* **Resource Classes**
- Specify the external data source:
  - *File name:* **My Excel for ARM Turorial.xlsx**
  - *Worksheet:* **Resource Classes**
  - (Optional) *Table:* **ResourceClasses**



- Label the block **Resource Defs**
- Click **OK** to close the Data Import Export block's dialog, then **Save** the model

Import the resource requirement definition to Resource Requirement Expressions

- Add a second **Data Import Export block** (Value library) to your ARM Tutorial 2 model.
- In the dialog of this Data Import Export block, leave the block's action as the default **Import from Excel spreadsheet** to **ExtendSim database**
- Specify the ExtendSim data target as:
  - *DB:* **Advanced Resources**
  - *Table:* **Resource Requirement Expressions**
- Specify the external data sources as:
  - *File name:* **My Excel for ARM Turorial.xlsx**
  - *Worksheet:* **Resource Requirement Expression**
  - **(Optional)** *Table:* **ResourceRequirementExpressions**
- Label the block **Requirements**
- Click **OK** to close the block's dialog, then **Save** the model

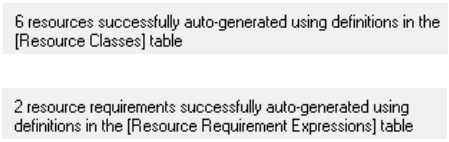


Generate the information

At this point the resources and resource requirements have been defined and the importing of those definitions has been set up. However, the resources and resource requirements still need to be imported and created.

As was discussed on page 32, there are two ways to create the resources and requirements: run the simulation or auto-generate. Before running the simulation, it is a good idea to see if everything has been imported correctly.

- In the Resource Manager block, click the **Auto-Generate Components** button located in the top right of the dialog.
- After clicking the button, you should get two messages:



☞ If Excel is not open when the Auto-Generate Components button is clicked, the ARM system will first launch Excel.

Auto-generation imports the definitions from Excel into the Resource Classes and Resource Requirement Expressions tables of the Advanced Resources database, creates the resources and the resource requirements, and makes them available for use in the model.

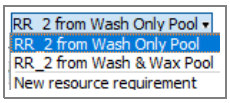
- Click OK to close the dialog of the Resource Manager block and save its changes

Verify the process

You can always look in the Advanced Resources database. However, the Resource Manager block also provides information.

- To examine resources, change the mode on the Resource Manager block's *Resources* tab to **Edit Resources**. Make sure all the resources you expect to be there are, and that they are properly configured.

- For the resource requirements, select the "Create/modify resource requirements" check box on the Resource Manager block's *Requirements* tab, then click the resource requirement popup menu.



Selecting the requirement in the queues

The items that arrive to a Queue can get their requirements from:

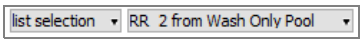
- A list of resource requirements
- A connector value
- An attribute value

For this model, you want to the queues to get the resource requirement from a list selection.

☞ See "Tutorial 3: Use Attributes for Flexibility and Scalability" on page 39 for how to use attributes to select which pool the desired resources are in; that is a more sophisticated approach than used for this tutorial.

Tell the Wash Queue where to get its resource requirement

- Open the dialog of the **Wash Queue**; notice that its behavior has already been set to use advanced resources
- In the section for defining the resource ordering behavior, choose to *Get resource requirement from:* **list selection**
- Choose **RR_2 from Wash Only Pool**



- Click OK to close the dialog and save changes

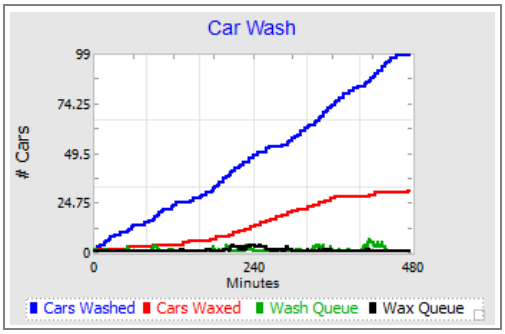
Tell the Wash/Wax Queue where to get its resource requirement

- Open the dialog of the **Wash & Wax Queue**
- In the section for defining the resource ordering behavior, choose to *Get resource requirement from:* **list selection**
- Select **RR_2 from Wash & Wax Pool**

- Click OK to close the dialog and save changes
- Save the model

Run the model

Running this model generates the same information as running the model where resources and requirements were created through the dialog of the Resource Manager block. The only difference is that the resources and resource requirements for this model were generated using the information entered into an Excel workbook.



☞ The final model, shown above, is named ARM Tutorial 2 Final and is located at Documents/ExtendSim/Examples/Tutorials/Advanced Resources.

Next steps

Now that you know the basics, look at your model and see if you noticed these issues:

- The Select Item Out block (Select Route) uses probabilities to route the cars. It is more likely that your model would assign attributes to the cars and route them using those attributes.
- Each Activity block needs a Queue in front of it for selecting the resource requirement for that line, and the requirement must be manually selected. This is fine for a small model but most models would have dozens or even hundreds of Activity blocks.

See the next chapter for a model that has one Queue and uses an item attribute both to route the cars and to tell the Queue which pool the resources should come from.

ARM
Tutorial & Reference

Tutorial 3: Use Attributes for Flexibility and Scalability

Overview

The previous chapters showed how to create resource requirements by explicitly specifying the values for resource properties and quantities. They also demonstrated how to explicitly specify which resource requirements to use when items arrive in a Queue block.

This chapter shows how to make ARM models more flexible and scalable by using attributes to implicitly specify those values.

- Part I, starting on page 41, shows how to have each item's attribute value specify which resource requirement it will use. This approach is especially useful when you have many pre-defined resource requirements. It also minimizes the complexity of the model structure since it requires fewer blocks.
- Part II, starting on page 43, demonstrates how an item's attribute values can specify the quantity of resources required as well as the value of a resource property. This approach minimizes how many resource requirements the model needs.
- Part III, starting on page 45, shows how to associate a property with an item attribute.

This chapter assumes that you already know how to use item attributes, including string attributes. If not, see the Items, Properties, and Values chapter of the User Reference.

Explicit methods

As you saw in the previous chapters, explicit specification methods work well for ARM models that have a small number of resource allocation rules. However, in most real-world models, it is necessary to have many different requirements for allocating resources. If explicit specification methods are used, different filtering conditions, quantity expressions, and resource requirements would need to be created to accommodate each difference, no matter how small the difference is. Also, a separate Queue block would need to be used for each resource requirement. Consequently, explicit specification methods can cause models to grow in size and complexity and become difficult to scale and maintain.

Implicit methods

Instead of explicitly specifying property values, resource quantities, and requirement selections, ARM allows users to use attributes to implicitly specify these values. Embedding attributes within resource requirement expressions enables ARM to look up the value of the referenced attribute whenever a resource requirement, or any of its internal components, are

being executed for an item. This significantly reduces the number of requirement expressions that need to be created in a model.

For example, the filtering condition that explicitly specifies the pool as being “Wash only pool” and the filtering condition that specifies the pool as being “Wash & Wax Pool” can be replaced with a single filtering condition that uses an item attribute to specify the property value of the pool. When a property value is associated with an attribute in a filtering condition, ARM will look up the current value of the specified attribute to determine which property value to use when the filtering condition is executed.

Using attributes to specify values

Attributes can be used to specify the values for any resource property in a filtering condition.

- If a filtering condition refers to a resource property that has a numeric data type, the numeric value of the attribute is used.
- If a filtering condition refers to a resource property that has a string data type, the value of the attribute must be a string attribute as discussed below.

String attributes

Attribute values are stored internally in ExtendSim as numbers. When an attribute is used to refer to an index within an enumerated list of string values, it is referred to as a string attribute.

String attributes are defined and associated with enumerated lists in the Executive block. The enumerated list of string values are specified in the attribute string values table in the Item Attributes tab of the Executive block. These values can be manually entered or they can be linked to a database table.

Whenever a string attribute is used to specify a string type resource property in ARM, it is strongly recommended that the enumerated list of string values be linked to a database table. For example, a filtering condition that uses a string attribute to specify the value of a pool should use a string attribute that is linked to the Pools database table in the Advanced Resources database table. Doing this automatically couples the enumerated list to the current list of pools in an ARM model. Any changes made to the list of pools would then be automatically updated in the string attribute’s enumerated list.

When a filtering condition uses a string attribute that is linked to an ARM database table, the value of the attribute is interpreted to be a record index in the linked database table.

How ARM handles item attributes

When items arrive to ARM queues, the queues send the Resource Manager block two pieces of information:

- The resource requirement
- The item index

The Resource Manager then attempts to satisfy the specified resource requirement by executing the expressions within the resource requirement. However, if any of these expressions contain a reference to an item attribute, the Resource Manager first replaces the reference with the actual value of the attribute.

Embedding attributes within resource requirement expressions means that ARM can look up the value of the referenced attribute whenever a resource requirement or any of its internal

components is being executed for an item. This significantly reduces the number of requirement expressions that need to be created in a model.

Part I: using an item's attribute value to select a resource requirement

The models in the two previous chapters had only two simple but specific resource requirements. Models that use ARM usually have multiple and more complex resource requirements that filter on compounded resource properties. For example, a requirement might be that the resources have a specific pool as well as a particular range of utilization.

Also, it may not be enough to specify at the beginning of the simulation which pool a resource comes from. Depending on what happens during the simulation, items might need flexibility in choosing a resource requirement.

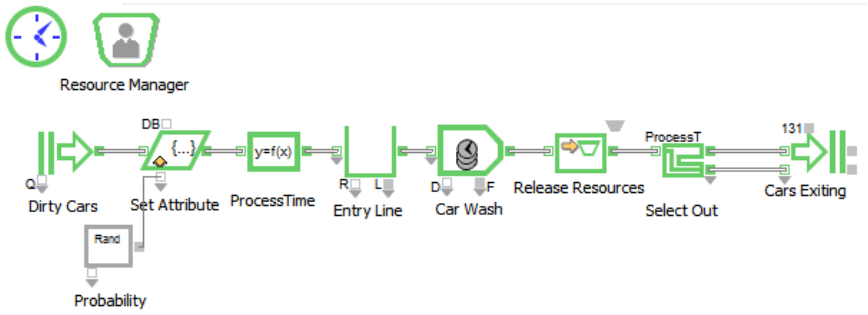
This tutorial shows how to use an item's attribute value to specify which resource requirement a Queue will use for that specific item.

Your task: use a scalable resource requirement

For brevity, the model already has one resource requirement that will use the value of an item's attribute to search for a related Pool value, selecting resources based on that property. Your task for this tutorial will be to let the Queue know that each item's attribute value will determine which resource requirement it has.

Open the model

- Launch ExtendSim
- Open the model **ARM Tutorial 3A** from the folder Documents/ExtendSim/Examples/Tutorials/Advanced Resources
- Save the model as **My ARM Tutorial 3A**

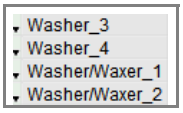


The models used in the ARM tutorials do not use the best modeling techniques. However, they are purposefully structured to make it easier for you to learn ARM concepts. For a scalable, logical example see the *Semi-Conductor Fab* model located at ExtendSim/Examples/Discrete Event/Resources and Shifts/Advanced Resource Mgt. It shows how to use ARM with 2D animation of a material handling system.

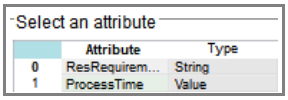
About the model

This model is similar to the ones used in the previous two chapters and the model assumptions are the same, including that:

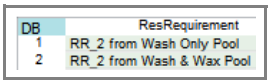
- There are 4 *Washer* resources and 2 *Washer/Waxer* resources
- There are two resource requirements, each of which requires two resources from specific pools: *RR_2 from Wash Only Pool* and *RR_2 from Wash & Wax Pool*



However this model is structured differently because, as seen in the Executive block, the items (cars) have a string attribute (ResRequirement) that uses string values to specify whether the cars will only be washed or also waxed.

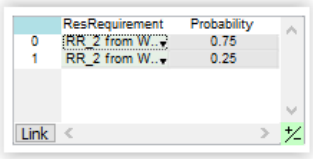


(As explained in "String attributes" on page 40, it is best practice to link the string attribute values to the requirements in the Resource Requirements database table as is done in this model.)



Items also have a value attribute (ProcessTime) with two values (6 or 8) depending on whether the car gets waxed or not, so the Activity (Car Wash) can tell how many minutes to process each car.

- As items are created, they are assigned the ResRequirement string attribute. The Random Number block, shown here, specifies that 75% of the items will have the string attribute value "RR_2 from Wash Only Pool" and 25% will have the string attribute value "RR_2 from Wash & Wax Pool".



- An Equation(I) block attaches the value attribute (ProcessTime) to each item. As seen in the equation, if the item's ResRequirement value is *RR_2 from Wash Only Pool*, the value of ProcessTime will be 6; otherwise it will be 8.

```
if (ResRequirement == 1)
   ProcessTime = 6;
else
   ProcessTime = 8;
```

- The Activity block (Car Wash) uses the value of each item's ProcessTime attribute to determine how long to process each car.

Set the Queue

Since the attributes have already been set, the only thing left to do is to update the Queue block. In the dialog of the Queue (Entry Line):

- Change the Queue's behavior to **advanced resource queue**
- Define the resource ordering behavior as: **Get resource requirement from: attribute value: ResRequirement**
- Click OK to close the Queue's dialog and save its changes
- Run the model

As the model runs:

- The Queue (Entry Line) will use each item/car's ResRequirement string attribute to determine if the cars are wash only or also wax.
- The Activity (Car Wash) will use each item/car's ProcessTime attribute to determine how long that activity will take.

The final model is labeled ARM Tutorial 3A Final; it is located at ExtendSim/Examples/Tutorials/Discrete Event/Advanced Resources.

Conclusion

In the previous chapters, the items' resource requirement was explicitly specified in each Queue's dialog. This meant that every item entering that queue had the same resource requirement. It also meant that you needed a separate Queue for each different resource requirement.

In this example, the Resource Manager determined the appropriate resource requirement for each item by evaluating the value of the item's ResRequirement attribute.

Note that the only additional work from how this model would be built using explicit resource requirements are:

- Creating a string attribute (ResRequirement) to represent the Resource Requirement
- Linking the table of string attribute values to the "Resource Requirements" table of the Advanced Resources database
- Choosing in the Queue that the resource requirement comes from the value of each item's ResRequirement attribute

Why the string attribute values are linked to an ARM database table

String attributes are managed in the Executive block's Item Attribute tab. The attribute values can be entered manually or, as was done here and seen in the Executive block's Item Attribute tab, dynamically linked to a database table.

Since each item in this model will use the value of its ResRequirement attribute to specify its resource requirement, the best approach is to link the table of attribute values to the database table for Resource Requirements. This protects your model's integrity in case you subsequently add or delete resource requirements. When the attribute values are linked to the database table, those attribute values will automatically and immediately change if any changes are made to the database table.

☞ To learn more about ExtendSim databases and linking data tables to database tables, see the separate document ExtendSim Database Tutorial & Reference.

Part II: using an item's attribute value for maximum flexibility

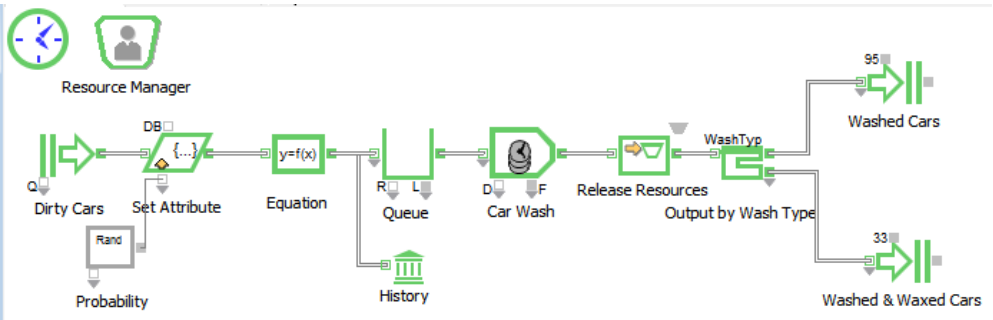
In earlier chapters you created two resource requirements, one for each Pool value. And those resource requirements had fixed quantities: 2 Washers from the Wash Only Pool or 2 Washer/Waxers from the Wash & Wax Pool.

What if you wanted to run the model with a requirement of 3 Washers from the Wash Only Pool or 1 Washer/Waxer from the Wash & Wax Pool? What if you had hundreds of pools, rather than just two?

You could create numerous resource requirements with different quantities of resources from different pools. Or you could create one flexible resource requirement that used each item's attributes to specify the pool and the quantity of resources wanted.

Open the model

Open the model **ARM Tutorial 3B Final** from the folder Documents/ExtendSim/Examples/Tutorials/Advanced Resources.



About the model

This model is similar to model 3A in that:

- 75% of the cars need only washing while 25% also need waxing.
- There are 4 Washer resources who have the Pool property value of "Wash Only Pool" and 2 Washer/Waxers who have the Pool property value of "Wash & Wax Pool".

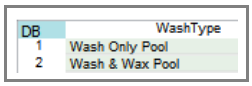
However, there are differences from previous models:

- This model only requires 1 Washer per car, rather than the 2 Washers required in the previous model.
- This model has two new attributes.

By using attributes to dynamically specify some of the variables in a resource requirement, this model shows how to reduce the number of resource requirements that have to be managed. Specifically, this model uses attributes to dynamically specify the required resource pool and the required quantity of resources from that pool.

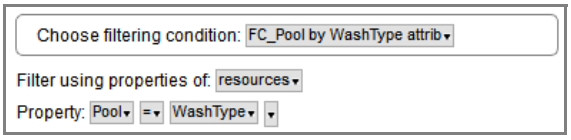
Steps that were needed

- Created two new attributes:
  - A string attribute named *WashType* with two string values: Wash Only Pool (numeric value = 1) and Wash & Wax Pool (numeric value = 2) as seen in the Item Attributes tab of the Executive.



DB	WashType
1	Wash Only Pool
2	Wash & Wax Pool

  - A value attribute named resQuantity used to specify how many Washers or Washer/Waxer resources are needed.
- Dynamically linked the WashType string attribute to the Pools table in the Advanced Resources database, as indicated in the screen shot of the Executive's Item Attributes tab above and discussed on page 43.
- Created a filtering condition that uses the WashType attribute to specify the resource pool. The filtering condition, "FC_Pool by WashType attrib", is displayed in the Requirements tab of the Resource Manager block. See page 45 for how this was done.

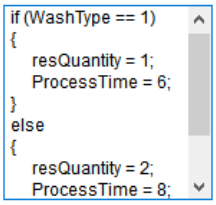


- Created a quantity expression that refers to the filtering condition, above, to specify the required pool and uses the resQuantity attribute to specify the required quantity of resources. The quantity expression "QE_resQuantity from Pool by WashType" is displayed in the Requirements tab of the Resource Manager block.

- Created a resource requirement that uses the above quantity expression. See resource requirement "RR_QE_resQuantity from Pool by WashType" in the Requirements tab of the Resource Manager block.

- Set the resource requirement in the Queue block by selecting the above resource requirement from list select popup in the block's dialog.

- Used the Random Number block (Probability) in conjunction with a Set Attribute block to specify the value of the WashType attribute; that attribute's value is used to specify the required pool—either Wash Only Pool or Wash & Wax Pool.
- Used an Equation block to conditionally specify the value of the resQuantity attribute based on the WashType. The value of the resQuantity attribute is then used to specify the number of resources required.



- Added a History block (Value library) to help verify that the model is working as expected—the items arriving to the Queue have the expected attributes for pool and resource quantity.

Conclusion

This model accomplishes the following:

- There is only once resource requirement needed in this model and it can be selected from the popup list in any Queue block that is set to be an advanced resource queue.
- The quantity of required resources can be changed either in the Equation block's dialog or in the clone of the equation as desired.

Externalizing inputs

For the ARM Tutorial 3B model the quantity of required resources is set in the dialog of the Equation block. While not done in that model, it is good practice to externalize values whenever possible, at least in the first stages of creating and testing a model. For example, you could use a Data Specs block and two Constant blocks (Value library) to externalize numbers for the Equation block.

Externalizing values ensures name tracking plus you can see on the worksheet which values are being input to the Equation blocks, instead of opening their dialogs.

Part III: associating a Pool property with an item attribute

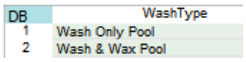
As a non-techie person I had some difficulty with this concept. So I thought it would be helpful to walk you through how to associate a Pool property with an item attribute.

☞ To compare to your work, a sample model named ARM Tutorial 3C Final is located at Documents/ExtendSim/Examples/Tutorials/Advanced Resources.

Create some resources and an attribute

To follow along, create a 3-block ARM model consisting of the Executive, the Resource Manager, and a Set block:

- Open a new model and place a Resource Manager and a Set block in it; both are from the Item library.
- In the Resource Manager block, create 4 *Washer* resources that have the Pool property value of "Wash Only Pool" and 2 *Washer/Waxers* that have the Pool property value of "Wash & Wax Pool".
- In the Set block, create a new string attribute named *WashType*.
- After you name the string attribute, the Executive block's Item Attributes tab opens. In the Executive, enter the string values *Wash Only Pool* and *Wash & Wax Pool*.

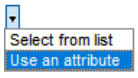


- Use the string value table's Link button to dynamically link the WashType attribute to the Pools table of the Advanced Resources database.
- Click the Save Changes button in the Executive's Item Attributes tab and close the dialog.
- Save your model.

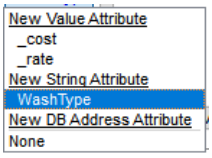
Create the filtering condition

- In the Requirements tab of the Resource Manager block, open the Filtering Conditions section and select the popup menu for **Choose filtering condition: new filtering condition**
- Choose to **Filter using properties of: resources** (the default)
- Since you want the item attribute to be associated with the Pool property, from the Property popup menu select **Pool**. At this point the expression should read as shown to the right.

- **Don't select a pool value yet!**
- Click on the arrow popup menu  to the far right of the expression. (In the above screenshot the arrow popup menu is to the right of the word Wash & Wax Pool.) As shown here, the options in the arrow popup menu are to *Select from list* (the default) or *Use an attribute.*



- Choose **Use an attribute**. Since you haven't selected an attribute yet, this causes the pool value popup menu to change to *None*
- Click on the word None (the pool value popup menu) and select the string attribute named **WashType**; this is the attribute assigned to each item before it enters the Queue block.



The expression should now read as "Property: Pool = WashType".

- The blue text for WashType indicates that it is an attribute
- "Pool" tells ARM that the Pool property will be associated with the WashType attribute—records in the Pools table will correspond to WashType attribute values.

▸ Click **Save As**, save the filtering condition as **FC_WashType Attribute**, and click OK.

☞ When a property value is associated with an item attribute in a filtering condition, ARM will use the current value of the attribute to determine which property value to use when the filtering condition is executed.

ARM
Tutorial & Reference

Reference

Overview

This section provides additional information for using the ExtendSim Advanced Resource Management (ARM) features.

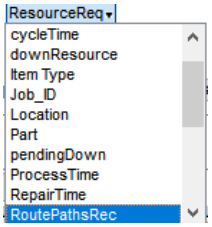
☞ For additional and up-to-date information, see the Help of the Resource Manager block.

Other example models

In addition to the Tutorial models, there are other more advanced ExtendSim models that show the use of ARM in real-world settings.

☞ The following models are located at ExtendSim/Examples/Discrete Event/Resources and Shifts/Advanced Resource Mgt.

Semi-Conductor Fab



This is a model with 2D animation of a material handling system. By using an attribute to specify the resource requirements for items, it demonstrates how to minimize the number of Queues required in an ARM model.

This technique enables the model to use a single ARM Queue for all of the different resource requirements to allocate resources from the 9 different types of tool sets in the model. Because this model is focused on presenting a 2D representation of parts moving and being processed in a factory, the actual model structure is logical in nature and focuses on the generic processing structure of requesting resources, waiting for them to become available, allocating them, processing jobs and releasing resources.

This generic processing is encapsulated in a single hierarchical block and demonstrates how to significantly reduce the complexity of an ExtendSim model's structure while still representing a highly complex manufacturing process.

Auto Club Service

This model demonstrates how to set up complex requirements to represent the flexible allocation of resources based on both resource availability and proximity to where services are being requested.

To accomplish this, the resource requirements in this model use compound expressions with OR conditions (see below) such that if the most desirable resources for a service request are not available, the model can attempt to use other specified resources in order to increase the overall throughput of the system.

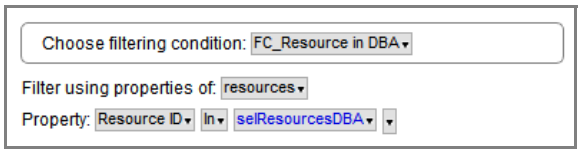
```
( 1 from Service Central AND 1 from Mechanic A Central ) OR ( 1 from Service North AND 1 from Mechanic A
North ) OR ( 1 from Service South AND 1 from Mechanic A South )
```

Dynamic Resource Qualification

This sophisticated model highlights a powerful feature—the ability to change, over the course of the simulation, which resources are available to perform certain tasks. This is accomplished by using database tables to store lists of the resources available for certain tasks, then modifying the lists dynamically during the simulation. The model shows how to deal with a situation in which resources in the same pool may or may not be qualified to perform the tasks at different times during a model run.

The model has a resource requirement that filters on resources listed within database tables. The tables contain lists of resources qualified to perform a specific task and are created by an Equation(I) block using the current system state and the properties of the items passing through.

The Equation(I) block sets a pointer to these tables in a DB Address attribute that is referenced within the resource requirement being used in the ARM Queue. It does this using a DB Address attribute to point to the list (database table). By allowing this type of filtering condition, you have huge flexibility to create open-ended resource requirements that can change on-the-fly.



The Equation blocks completely control which resources get into the database table used as a filtering list. For example, in the Filtering Condition shown here, *selResourcesDBA* is the DB Address attribute that points to a database table that contains a list of resources. Because the "IN" clause is used, this filtering condition will only allow ARM to look at resources in the database table pointed to by this list. Resources are referenced as record indexes in this list table.

☞ See also “Sorting criteria” on page 79.

Comparison of explicit resource modeling methods

As discussed in the User Reference, there are several ways resources can be modeled—explicitly, implicitly, and conceptually.

You can model resources explicitly using Resource Items, Resource Pools, or ARM. A summary of the differences between those methods are given in the table below.

- For a complete description of the similarities and differences between the ARM, Resource Item, and Resource Pool methods, see the Discrete Event: Resources and Shifts chapter of the User Reference.
- Also see the list on page 2 of this document describing when and why you should use ARM.

☞ The ARM system can co-exist with the other resource modeling approaches (e.g. items as resource items and non-ARM resource pools). Thus ExtendSim models can use a mixture of resource modeling methods.

Capability	Resource Items	Resource Pools	ARM
Dedicated user interface integrated with an ExtendSim database			X
Resources have properties	X		X
Resources can be fractional		X	X
Resources can be shared across multiple items	X		X
Resources know which items are waiting		X	X
Remote resource allocation		X	X
Requires connections between blocks	X		
Logging of statistical results, transactions, etc.			X

Resources

A *resource* is a stock or supply of something that can be drawn on as needed for an organization to function effectively. Resources are both the factors required to accomplish an activity and the means by which process activities and operations are performed. Examples of resources include equipment, personnel, space, energy, time, information technology, and money.

Use in simulation models

In models, items will sometimes require resources before they can proceed to the next step in a process. For example, a car might need an attendant to drive it through the car wash or a piece of equipment might need to be assembled by a robot. Resources provide a service to the items in a model; their lack of availability causes constraints on the flow of items.

One of the main reasons to model a process is to determine if there is sufficient resource capacity to meet the desired performance of that process. Simulation models provide the means to analyze how and when resources are being utilized and what impact resource availability has on process performance. For example, a simulation model might be used to observe resource utilization and availability over time in order to discover what the most effective resource levels are to achieve the desired process performance level. It also can be used to study the impact of increasing or decreasing demand on an existing system. So you might try to improve resource utilization without causing overly long waiting lines or determine how to reduce waiting lines without adding more resources.

ARM resources

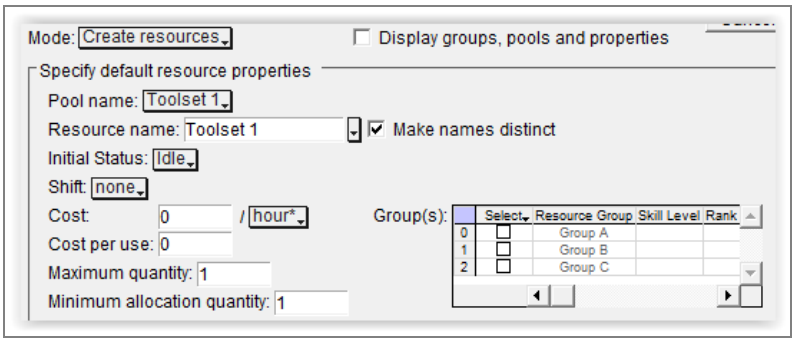
As discussed on page 61, resources used in the Advanced Resource Management (ARM) system have properties that distinguish them from each other. And they can be organized into groups based on common sets of properties, as discussed on page 56. In the Advanced Resources database, resources are represented as records in the Resources table and the fields in that table represent the different properties that a resource has.

☞ In ARM every resource must be a member of one, and only one, pool. However, based on its skill set or rank, a resource can be a member of multiple groups.

Creating resources

Resources for ARM can be defined and created:

- In the Resources tab of the Resource Manager block (Item library), as shown at right and described in the tutorial that starts on page 19



- Using an Excel worksheet, as shown in the tutorial that starts on page 31
- Directly in the Resources table of the Advanced Resources database

☞ Since you can't specify the properties of resource in the Resources table, it is strongly suggested that you use one of the first two methods rather than creating resources directly in the Advanced Resources database.

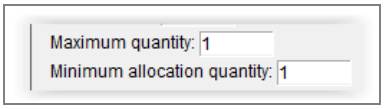
When you create resources, you also specify default resource properties—which properties all the resources in a specific pool will have. Once the resources have been created, you have the option to change the properties of individual resources.

Fractional and multiple resources

When resources are created, each is by default one resource as indicated by its Maximum Quantity property. However, the ARM system also supports:

- *Fractional resources*, where each record in the Resources table can represent a fraction of a resource. This allows parts of one resource to be allocated to different items. For example, fractional resources could be used to represent a multi-tasking situation. A worker might be assigned multiple tasks concurrently where each task would require a different fraction of the worker's total capacity.
- *Multiple resources*, where a number of resources with an identical set of properties are represented in a single record in the Resources table (note that there is a reduction of fidelity with this option).

Two resource properties control fractional and multiple resources.



The *Maximum Quantity* property determines whether the record in the Resources table represents one resource, a fraction of a resource, or multiple resources. This property is used by ARM to manage the quantities that resources make available to the model—how many resources, or what fraction of a resource, is represented by one record in the database. By default, this property is set to 1, indicating that the resource is a single, whole resource. Changing that number to a fraction or to a number higher than 1, results in fractional or multiple resources, respectively

Correspondingly, the *Minimum Allocation Quantity* property represents the smallest quantity of a resource that can be allocated to a resource order. It is used by ARM to control how resource quantities are allocated to items. By default this property is set to 1.

☞ Each resource's Minimum Allocation Quantity must be less than or equal to its Maximum Quantity.

The Available Quantity is a read-only field in the Resources database table that reports the amount of the Maximum Quantity that is currently available for allocation to resource orders.

How ARM uses resources in the simulation

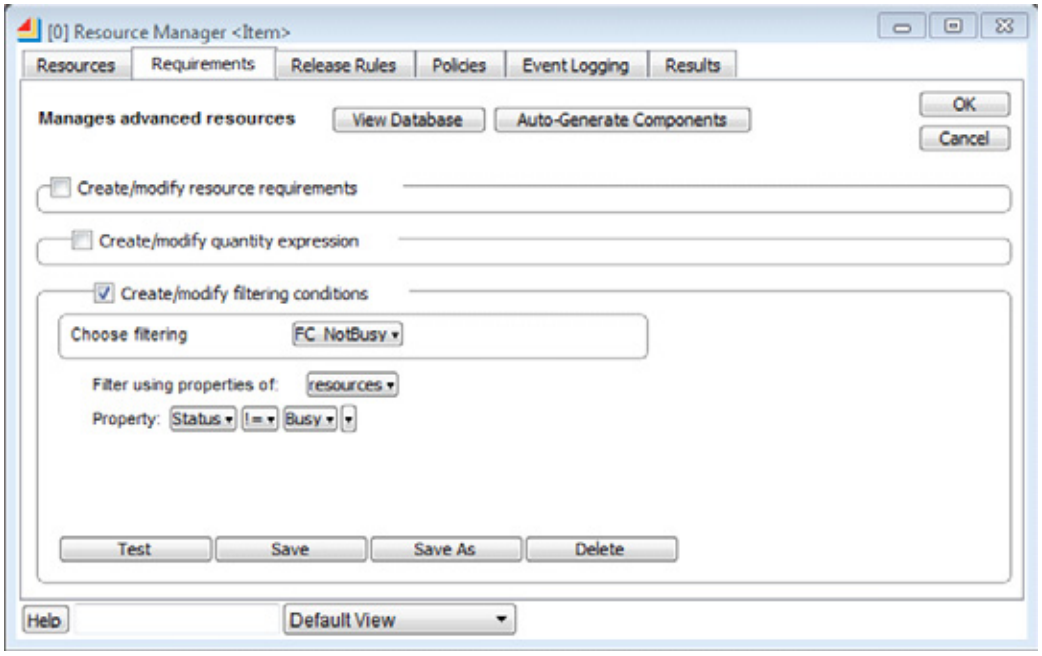
As the simulation runs, ARM uses internal policies such as resource recycling and resource sharing.

Resource recycling

Resource recycling allows an item that is requesting resources to use *busy* resources that have already been allocated to the item to fully or partially satisfy its resource requirements. Whenever ARM receives a request by an item to satisfy a resource requirement, it will attempt to use any of the *busy* resources allocated to that same item before trying to obtain *idle* resources. For example, if a resource requirement needs 1 resource from Pool A and 1 resource from Pool B and the item requesting to have this requirement satisfied already has 1 resource from Pool A allocated to it from a Queue, when it gets to the next Queue ARM will only need to allocate 1 additional resource from Pool B in order to satisfy the resource requirement.

⚠ Resource recycling is the default mode for ARM resource allocations.

To prevent ARM from recycling *busy* resources, add a filtering condition to resource requirements that prevents resources in the *busy* state from being considered for the resource requirement. See an example of how to create this filtering condition below.



Resource sharing

Resource sharing allows individual resources to be shared across multiple items at the same time. Resources are shared by cloning (Unbatch block) or spawning (Query Equation block) one or more items that are already associated with ARM resources. For each cloned/spawned item, ARM creates a new resource order. These new resource orders become a copy of the original resource order so that whatever resources were associated with the original resource order become associated with new resource orders.

Resources are shared by spawning items from an inbound item with attached resources in a Query Equation block or copying items from an inbound item with attached resources from an Unbatch block in non-preserve uniqueness mode.

When resources are shared using the Query Equation block, ARM copies the list of resources associated with the inbound item to the resource orders created for each spawned item. If there is a pass through item, ARM preserves the resource order for that item. Otherwise, ARM destroys the resource order associated with the inbound item and its *resource order-item-resources* relations.

When the Unbatch block is used, ARM copies the list of resources associated with the inbound item to the resource orders created for each cloned item and destroys the resource order associated with the inbound item and its *resource order-item-resources* relations.

When resources are shared across multiple resource orders, they remain in a *busy* state until they have been explicitly released from all of the items they are associated with. ARM uses the [Shared Count] field in the [Resources] table to keep track of how many times a resource has been shared across items in a model. A shared count of 1 indicates that a resource is being shared with one item and is thus associated with two items.

Resource reassignment

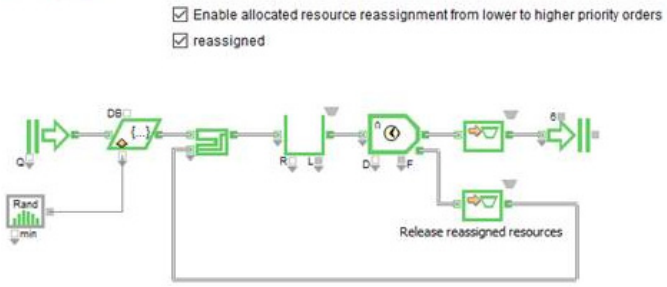
Resource reassignment allows ARM to reassign allocated resources from lower priority items that are blocking higher priority items from having their requirements satisfied. Resource reassignment is enable by checking the “Enable allocated resource reassignment from lower to higher priority items” checkbox in the Policies tab of the Resource Manager dialog.

If resource reassignment is enabled, whenever ARM is unable to satisfy the requirement of an item’s resource order, it will search for allocated resources that are blocking the resource order’s requirement from being satisfied. The conditions for resource reassignment are as follows:

1) An item (the waiting item) is unable to have its resource requirement satisfied
2) The resources that are blocking the item from having its requirement satisfied are allocated to items that have lower priorities than the waiting item
3) The items that have allocated blocking resources reside in Activity blocks
4) These Activity blocks have Preemption enabled for resource reassignment from the Resource Manager block.

When ARM is able to satisfy the 4 resource reassignment conditions, it sends messages to all of the Activity blocks having resident items with blocking resources allocated to them. This triggers the Activity blocks to release items with blocking resources and causes those items’ attached resources to have their status changed to “reassigned”.

To make these resources available to the higher priority items, the modeler must explicitly release the reassigned resources from the preempted items. The preferred method is shown below.



Pools

A pool is a required property of a resource. The pool property allows modelers to refer to a collection of resources by a common organizational relation. ARM uses the Advanced Resources database to explicitly represent and keep statistics for all of the distinct pools that have been defined in a model.

☞ Every resource must have one (and only one) pool as one of its properties.

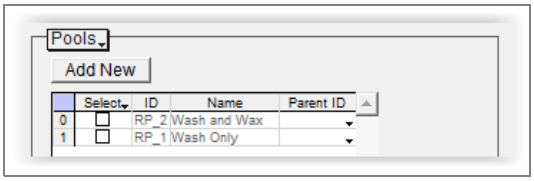
Except for the Skill Level and Rank properties, all the properties of a resource are associated with a particular resource pool, as discussed on page 65. Pools can be grouped hierarchically in a parent/child relation. Statistical information for pools, such as the total number of orders serviced and utilization, are provided in the Resource Manager block’s Results tab.

To understand the link between Pools and resources, notice that the Pools database table is a parent to the Resources table's Pool field.

Checkbox to display groups, pools, and properties

When the Resources tab is in Create or Edit mode, a *Display groups, pools, and properties* checkbox appears at the top of the tab. You can create a new pool when you define a new resource in the Resource Manager's Resources tab. Or you can create multiple pools by checking that box.

When this box is checked, a secondary frame appears to the right with a popup menu for selecting Pools, Groups, or Properties.



For each of those, you can add new or delete those that are selected. For example, this is where you could add custom resource properties.

☞ If you delete a Pool that already has resources, the resources are also deleted.

Groups

A group is an optional method for organizing resources. Each group consists of a collection of one or more resources where the resources can belong to the same or different pools. A resource can be a member of none, one, or multiple groups.

A resource's properties are represented by field values in the Resources table in the Advanced Resources database. Because each record in this table has only one pool property value, each resource record and consequently, all of the property values associated with that record, can only be associated with one resource pool.

In many situations however, the capability of a resource needs to change depending on how it is being used. In particular, a resource might have a different skill level depending on the type of task it can perform. Groups are a cross-pool method for organizing resources based on specific criteria, such as skill level. In the Resources table, the Group field holds the group information. The Group Resources table lists all the resources per group.

Groups are another way to control which resources get allocated to items. This is done by creating filtering conditions in resource requirements for resource groups and group properties. Groups provide modelers with a mechanism for:

- Organizing collections of resources by the type of task they are capable of performing
- Varying the skill levels of resources as a function of the tasks they are performing

As discussed more on page 66, groups have properties and the resources in a group are distinguished by two group-level properties: 1) skill level and 2) rank.

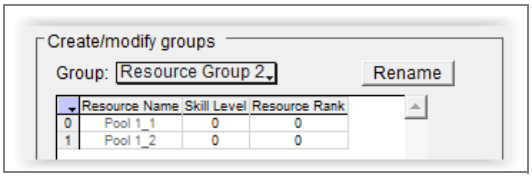
☞ Although a resource can only belong to only one pool, it can be a member of many different groups.

Creating groups and grouping resources

As mentioned in "Checkbox to display groups, pools, and properties" on page 56, you can create new groups when the Resource Manager's Resources tab is set in *Create resources* mode. Another method is to change the mode to *Manage groups*, as discussed below.

To create a new group and include some resources in that group:

- Go to the Resources tab of the Resource Manager block
- Select **Mode: Manage groups**
- Optional: in the *Set resource filters* frame at the top, choose filters to limit which resources are displayed
- In the *Create/modify groups* frame at the right:

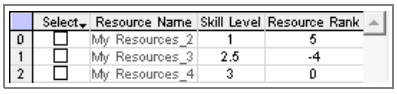


  - Choose **Group: New resource group**
  - Name the new group
  - In the dialog that appears, choose whether or not the system should create a default resource requirement for the group. This option is explained in “Default resource requirements” on page 76.
- In the *Select resources to group* frame on the left:
  - Check the Select box for each resource you want to include in the group
  - Click the arrow button (shown at right) that is located between the two frames (*Select resources for groups* and *Create/modify groups*) at the bottom of the dialog. Clicking the button causes the selected resources to become members of the group.

☞ Like resources (see page 31), groups can also be defined in Excel. When group definitions are imported into the Resource Classes database table, ARM automatically creates the groups using information contained in the Group, Skill Level and Rank fields.

Assigning properties to groups

As discussed on page 66, each member of a group can be assigned a Skill Level and/or a Resource Rank. When a group requirement is executed, these group properties can be used to filter and prioritize which resources to select from the group.



☞ Any real number can be assigned to a group member as its Skill Level or Resource Rank.

Deleting groups or their members

- Go to the Resources tab of the Resource Manager block
- Select **Mode: Manage groups**
- Optional: in the *Set resource filters* frame, choose filters to limit which resources are displayed
- In the *Create/modify groups* frame on the right, choose the group you want to delete
  - To delete the entire group, click *Delete Group*
  - To remove only certain members, check the appropriate Select boxes and click *Remove Selected Members*

☞ Deleting a group or removing individual members from a group, does not delete the resources. Furthermore, you can’t delete a group if it is referenced by an existing filtering condition. Before deleting a group, remove all filtering condition references to the group.

Using a group in a resource requirement

A filtering condition can be created that filters on a group property, either Skill Level or Resource Rank. That filtering condition can then be used as a component of a quantity expression, but *only* in conjunction with a second filtering condition that filters on a specific resource group. For more information, see “Using group filtering conditions” on page 74.

Status

A resource’s *status* tells which particular state the resource is currently in and thus whether the resource is available for allocation. The status is also used to record the time spent in a particular state; the time information is recorded in the Resources table of the Advanced Resources database.

Each resource is given an initial status of either Idle (available for allocation to an item) or Disabled (the resource is not currently used by the model). There are several additional states a resource can be in while the simulation is running and (as discussed below) the resource can *transition* back and forth between any of the states during the run.

Status settings

At any given time, resources are either available or unavailable to be allocated to items. ARM uses four general categories (Idle, Allocated, Down, and Disabled) to represent whether a resource is available or unavailable. Idle resources are available for item allocation while Allocated, Down, and Disabled resources are unavailable.Within these categories, there are 9 distinct status values as indicated below:

1) Available
    - Idle
2) Unavailable
    - Allocated: Busy, Reassigned, Reserved
    - Down: Off-Shift, Scheduled Down, Unscheduled Down, Failed
    - Disabled (disabled resources not currently used by the model)

The Resource Manager’s Results tab gives information about the amount of time each selected resource was busy, idle, and so forth.

By default, ARM will attempt to recycle any Busy resources that have already been allocated to a particular item to satisfy any new requests for resources by that item. To override recycling behavior, create a filtering condition that excludes Busy resources from being selected by an item’s resource requirement. See recycling on page 53.

Down status values

Resources can have a Down status due to:

- Being Off-Shift due to their association with a Shift block
- Experiencing a Scheduled Down or Unscheduled Down event due to logic being managed by the user
- Being in a Failed state due to their association with the Shutdown block

Status transitions

During a simulation run, resources continuously transition between the nine different status values discussed above. With few exceptions each status can transition to any of the others.

The result is that there are over 100 possible resource status transitions: Idle to Busy, Busy to Off-Shift, Off-Shift to Down, and so forth. These status transitions are the mechanism ARM uses to represent resource behavior.

Status transition events

Status transitions occur when an *event* occurs that causes a resource's status to change. There are a number of events that can cause an ARM resource to change its status:

1) Allocation events—a resource is allocated to an item.
2) Down events—a resource goes down for any one of several reasons. For example, it goes off-shift, fails during an operation, is taken down for scheduled maintenance, etc.
3) Reassignment events—a resource becomes reassigned from its current resource order to a higher priority resource order.
4) Disabling events—a resource is disabled from the current simulation run. For example, a piece of equipment is brought off-line to ramp down production capacity due to decreasing demand.
5) Enabling events—a resource is enabled, such as when a piece of equipment is brought online to ramp up production capacity in order to meet increasing demand.

Transition timing

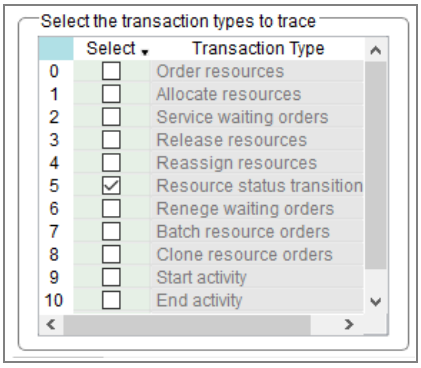
A transition is the result of a status transition event. The timing of when a resource's status value changes depends on the status of the resource when the event occurs.

- Some status transitions can occur at the same time as the associated event. These are referred to as *immediate status transitions*. An example of an immediate status transition is a resource that is in an Idle state becomes allocated to an item. In this case, the resource immediately transitions from Idle to Busy.
- Other status transitions must wait until the event associated with the current resource status expires. These are referred to as *pending status transitions*. ARM uses the pending status property of a resource to postpone its status transitions until the current status expires.

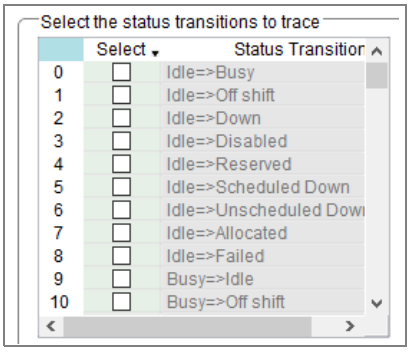
Logging status transitions

To facilitate the verification and validation of models, status transitions can be logged during a simulation run. To do this:

- Go to the Resource Manager block's Event Logging tab and check the box to *Enable resource transaction logging*.
- In the Transaction Types table check the box for *Resource status transition* as shown here. This causes the Status Transitions table to appear on the right side of the dialog. It is shown below.

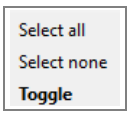


- Check the boxes in the Status Transitions table as desired, causing ARM to log status transition event information into the Resource Transaction Log table during the simulation run.



As seen in the Status Transition table there are over 100 possible resource status transitions.

Each status transition consists of a from status (the status a resource is currently in) and a to status (the status the resources is transitioning to).

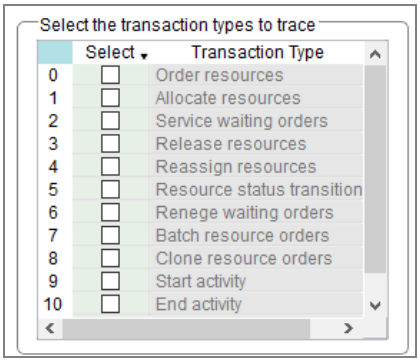


You can check individual boxes or use the down arrow in the Select header to choose all, choose none, or toggle between those you've checked and those that haven't been checked.

☞ Use logging judiciously. Choosing all the transitions can significantly slow simulations.

Transactions

The Event Logging tab of the Resource Manager block is for specifying which types of transactions to log during a simulation run. By default the information is gathered over the entire run, but the tab provides fields where specific start and end times can be entered. You can also choose that the table uses resource names instead of resource ID values.



There are several types of transactions that can be logged, as seen in the screenshot at right. Selecting *Resource status transitions* causes a second table to appear on the right, as discussed in "Logging status transitions" on page 59.

Click the desired Transaction Types or use the down arrow in the Select header to *Choose all*, *Choose none*, or *Toggle*. Selecting the Toggle option changes the checked boxes from those you selected to those you didn't select, and back again.

Whichever choices have been made in this table and in the Status Transitions table will cause simulation data to be written to the Resource Transaction Log table in the Advanced Resources database. Each transaction is represented by a separate record in that table.

☞ Use logging judiciously. Choosing all the transaction types can significantly slow simulations.

Transaction types

The Order Resources, Allocate Resources, and Release Resources transactions are most commonly selected. The details logged for these transactions are helpful when verifying that the proper resources are being allocated to and released from items and at the correct time.

Transaction Type	Description
Order resources	Occurs whenever an item arrives at a queue block (set to advanced resources mode) and makes a request for resources. When this transaction occurs, a resource order ID is assigned to the item. This transaction type only occurs once for each resource order ID.
Allocate resources	Occurs at the time resources are allocated to an item. When the resource requirement associated with an item's resource order can be satisfied and the item is able to depart from the queue it is in, the required resources are allocated to the item.
Release resources	Occurs at the time an item passes through a Resource Pool Release block set for ARM. For this transaction, the specified resources being released transition to their pending status.
Service waiting orders	Occurs every time an attempt is made by the ARM system to satisfy the resource requirement of a waiting item. This transaction occurs when items first arrive in a Queue block and whenever resources transitioning to the idle state attempt to become allocated.
Reassign resources	Occurs whenever resources that have been allocated to an item currently in an Activity block get reassigned to a higher priority item that is waiting for resources.
Resource status transitions	Transitions occur any time the status value of a resource changes. A status transition consists of a *from* status (the status a resource is currently in) and a *to* status (the status the resources is transitioning to).
Renege waiting orders	Occurs whenever an item is reneged from a queue block before resources have been allocated to it.
Batch resource orders	Occurs whenever items that have allocated resources are batched into new items.
Clone resource orders	Occurs whenever items with allocated resources pass through Unbatch or Query Equation blocks and new items are created.
Start activity	Occurs whenever items with allocated resources start their delay within an Activity block.
End activity	Occurs whenever items with allocated resources end their delay within an Activity block.
Failed to service orders	Occurs whenever an attempt to satisfy the resource requirement of a waiting item is unsuccessful due to the unavailability of required resources.

Properties

Individual resources, pools, groups, and items each have their own *properties*. Properties provide a means to distinguish between them and characterize their behavior. As the simulation runs, information about properties is displayed in the Resource Manager's Results tab.

Resource properties

Resource properties are used to distinguish resources from each other, control how resources behave, and record the history of how resources have been used during a simulation run.

Resource properties are represented as fields in the Resources table of the Advanced Resources database.

Resource properties are either *pre-defined* (discussed below) or *custom* (discussed on page 64).

Pre-defined resource properties

The ARM system provides over 40 pre-defined resource properties which are either ARM-managed or user-managed.

- *ARM-managed* pre-defined resource properties can only be set and modified using ARM-controlled interfaces, such as the dialog of the Resource Manager block; they cannot be directly modified in the Resources database table. They are further divided into input and output properties as shown in the tables below.
- *User-managed* pre-defined resource properties are managed during the simulation run directly by the user through an external interface such as the Shutdown block (Item library). See page 64.

ARM-managed pre-defined resource properties

These tables list the pre-defined input and output resource properties that are managed directly in the dialog of the Resource Manager block.

Input Resource Properties (ARM-Managed)

Resource ID (not modifiable)	Cost Per Unit Time
Name	Cost Time Unit
Pool	Cost Per Use
Shift	Initial Status
Maximum Quantity	Skill Level
Minimum Allocation Quantity	Rank
Available Quantity	Shareable

☞ The *Resource ID* property is automatically assigned to each resource. It is not modifiable.

Output Properties (ARM-Managed)

Total Orders Serviced	Total Cost
Total Idle Time	Total Failed Time
Total Busy Time	Total Quantity Allocated
Total Busy Off-Shift Time	Total Quantity Allocation Time
Total Reserved Time	Total Reassigned Time
Total Down Time	Total Scheduled Down Time
Total Off-Shift Time	Total Unscheduled Down Time
Total Disabled Time	Quantity Utilization

Output Properties (ARM-Managed)

Total Allocated Time	Utilization
Status	Resource Order ID
Status Start Time	Reassigned Resource Order ID
Pending Status	Shared Count
Pending Status Start Time	Off Shift

As shown on page 20, the first step in creating a resource is defining the input properties that the entire pool of resources will have—which pool the resources are in and what other properties the resources will have by default. When the resources are actually created, those default properties can be overridden for each resource. During the simulation run, ARM manages the values of the input properties and calculates the values of the output properties. Both types of properties are made available when creating or modifying filtering conditions.

- The Initial Status can be either Idle or Disabled, as discussed in "Status" on page 58.
- Shift is only available if the model contains at least one Shift block.
- Groups allow resources from one or more pools to be organized according to some commonality; they are discussed on page 56.
- The Maximum Quantity is used by ARM to manage the quantities that resources make available to the model—how many resources, or what fraction of a resource, is represented by one record in the database. The Minimum Allocation Quantity is used by ARM to control how resource quantities are allocated to items; this amount must be equal to or less than the Maximum Quantity. These two properties support the use of fractional and multiple resource features in ARM models, discussed on page 52.

User-managed pre-defined resource properties

The following resource properties are integrated with ARM functionality but are managed directly by the user. Instead of using the ARM interface, the modeler uses an external interface such as the Shutdown block (Item library) to manage these properties during the simulation run.

User-Managed Properties	Description
Scheduled Down	This Boolean checkbox field provides a means to change the status of resources during a simulation run through non-standard ARM mechanisms. If the box is checked during the run, ARM updates the status of the resource to be “scheduled down”. If unchecked, ARM transitions the associated resources into their current pending status.
Unscheduled Down	See description for Scheduled Down, above
Failed	See description for Scheduled Down, above
Failure Progress Type	Used by the Shutdown block to determine how the resource progresses towards its next failure event. Progress types are either *time-based* or *usage-based*.
TBF	Time between failures. Used when ARM is coupled with the Shutdown block, which uses this field to control the behavior of unscheduled down events for ARM resources.
TBF TTR Down Interruption Policy	This field is used by the Shutdown block to control the behavior of unscheduled down events for ARM resources. This property specifies how the Shutdown block will respond to an interruption of the progress towards the next failure event or the current in-process repair, depending on which state the resource is in. The options are: 1) preserve the current progress, 2) ignore the interruption, or 3) reset the progress.
TTR	Time to repair. Used when ARM is coupled with the Shutdown block, which uses this field to control the behavior of unscheduled down events for ARM resources.

Custom resource properties

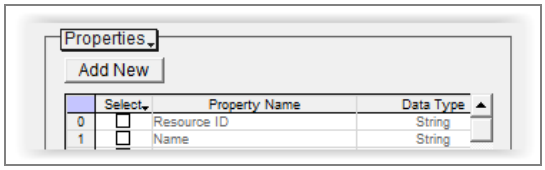
In addition to the pre-defined ARM resource properties, users can create custom properties and add them to the list of resource properties. These properties are user-managed and can be modified before and during a simulation run by directly accessing the Resources database table.

Custom properties can be added and modified:

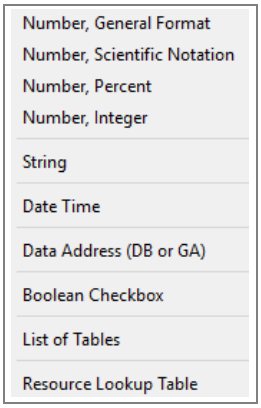
- Through the interface of the Resource Manager block, as shown below
- By defining them in an external application such as Excel, importing the definitions to the Resources database table, and auto-generating them
- By directly accessing the Resources database table (Advanced Resources database).

A custom resource property can have relation with a parent field in an ARM database table. For example, a custom property could be the location of a resource, while a Locations table in the Advanced Resources database could contain a list of locations. To relate the custom property to the Locations table, you can choose to make the custom property a child field with the Locations table as the parent. This is good database practice as it ensures the data is normalized, i.e., defined in one place only.

To create a custom resource property using the Resource Manager block's interface:



- In the Resource Manager's Resources tab, choose either **Mode: Create resources** or **Mode: Edit resources**
- Check the box to **Display groups, pools, and properties**
- In the new frame that appears to the right side of the dialog, change its mode from Groups to **Properties**
- Click **Add New** and name the new property
- Select a data type (number, string, etc.) from the popup menu, shown here
- Decide if the property should have a relation to a parent field in the database



The new property is now available for any resource. In the Resource Manager block it will be listed in the Resource tab's Properties table (when Display groups, pools, and properties is checked) and displayed in the Property popup menu when a resource filtering condition is created (Requirements tab).

☞ Custom resource properties can also be created directly in the Resources database table, or defined in an external application such as Excel and imported into the model for auto-generation.

Editing resource properties

To edit resource properties using the Resource Manager block's interface:

- Choose *Mode: Edit resources* in the Resource Manager's Resources tab.

In this mode, the tab is divided into two sections——the top is for narrowing the list of resources that will be displayed in the table that is at the bottom.

- For individual resources, change the appropriate settings in that resource's row
- To change one or more properties for several resources, in the *Edit filtered resources* section:
  - Choose whether column heading selections should apply only to the selected rows or to all rows
  - If *selected rows* is chosen, check the boxes in the Select column for the resources to be changed
  - Use the popup menu in the column heading to choose the new setting. For example to change all of the displayed resources to the disabled state, choose that setting in the Initial Status column.

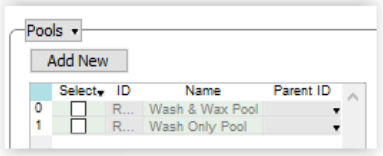
☞ The fields in the Groups column are only for information; they cannot be changed.

Pool properties

A pool can have an *ID*, a *Name*, and a *Parent ID*. The ID is assigned by the ARM system. The Name (which must be unique) and the Parent ID are user-modifiable properties. Pools can be

structured hierarchically; if one pool is part of another pool, the Parent ID identifies the next highest pool in that hierarchy.

To see a model's pools and their properties, go to the Resource Manager block's Resources tab and check the box to *Display groups, pools, and properties*. Then choose *Pools* from the popup menu in the table that appears on the right side of the dialog.



For a summary of each pool's results from the simulation run (total idle time, total utilization, and so forth), see the Resource Manager block's Results tab or the Pools table of the Advanced Resources database.

Group properties

The group properties are *Name*, *Skill Level*, and *Resource Rank*, all of which are user-modifiable. While they are considered group properties, the Skill Level and Resource Rank are assigned to the individual members (resources) of the group.

Group properties are useful when creating or modifying filtering conditions. Both the Skill Level and the Resource Rank are used to prioritize which resource to select first from the group, when a group requirement is executed. The Skill Level or Resource Rank can be any real number.

The Groups table of the Advanced Resources database lists all of the model's groups. The Group Resources table displays all of the groups in the model as well as the properties and Resource ID of each of their members. (Remember, a resource can be included in more than one group.)

For information on creating and managing groups, see "Groups" on page 56.

☞ While the principle is the same, a group's Resource Rank is not the same as a resource's Resource Requirement Rank.

Item properties

Every item entering a Queue set for advanced resources has a unique *Resource Order ID* (_AR Order ID property) assigned to it. This is an item property the same as attributes and priority; item properties are discussed in the Discrete Event section of the User Reference.

The ID refers to a record index in the Resource Orders table of the Advanced Resources database. (See "Resource orders" on page 66.) When Resource Pool Release blocks in ARM mode release all resources from a Resource Order item, the _AR Order ID property is set to Blank. When an item passes through multiple ARM-based queues before releasing its resources, the _AR Order ID will be the last property assigned to the item but ARM will keep track internally of all the assignments.

While not an item property, resource requirements have a *resource requirement rank* that controls the order in which idle resources search for waiting items to service. A resource requirement with a better rank will have first choice of any available items that require that resource. For more information, see "Resource allocation policy" on page 78.

Resource orders

If each item could only have one resource requirement, and if each resource requirement could only be selected by one item, statistical information could be obtained just from the resource

requirement's usage. Since that's not how the real world works, the ARM system allows each item to have multiple resource requirements and even the same resource requirement multiple times but at different Queues. Additionally, several different items can each have the same resource requirement. To track data and calculate statistical information, the ARM system uses resource orders.

Definition

A *resource order* represents a specific request for resources by an item. The information associated with a resource order is represented as a record in the Resource Orders table of the Advanced Resources database. Whenever an item makes a request for ARM resources, a resource order is created and the record index of the resource order in the Resource Orders table is assigned to the item's Resource Order ID (_AR Order ID property) as discussed in "Item properties" on page 66).

Each resource order is associated with only one resource requirement. The resource requirement specifies which types of resources and how many of them are required by the item and thus, by the associated resource order. When the required resources for a resource order become available, ARM allocates the resources to the resource order.

A resource order remains associated with an item until all of the resources allocated to the order have been released. Because items can make and accumulate multiple requests for resources, they can be associated with multiple resource orders at the same time.

Process

When an item gets to a Queue, the Queue makes a request to the Resource Manager for the resources specified by the item's resource requirement. The Resource Manager then:

- Converts that request into a resource order
- Creates a corresponding record in the Resource Orders database table referring to the resource request and identifying which queue the order came from, when it was ordered, and so forth
- Assigns a Resource Order ID to the item
- Attempts to satisfy the item's resource requirement
- When a resource is successfully assigned to an item, the Resource Order ID field in the Resources table is updated. That information is used to track the resource's last allocation. When an item is accumulating multiple resource orders, ARM creates a hierarchy of the resource orders created for the item. The _AR order ID property on the item is always the ID of the last resource order accumulated by the item.

This architecture allows the ARM system to track data and calculate statistical information such as when a resource was requested and the average length of time it took to satisfy.

For more detailed information, see "Primary transactions for ARM" on page 83.

Resource requirement overview

Resource requirements are a necessary component of the mechanism for allocating resources to items. The resource allocation mechanism consists of the following three components:

1) A requester—an item
2) A requesting & holding location—a queue block in advanced resource behavior
3) A specification of the required resources—a resource requirement

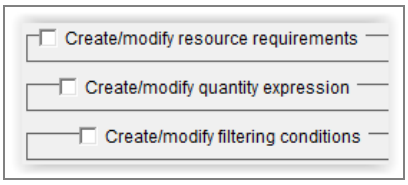
Resources are requested by items when they arrive in queue blocks which have their behavior set for advanced resource. The items are held in the queue blocks until the resource requirements are satisfied.

Definition

A resource requirement is a set of instructions that specifies how many of what types of resources to allocate to an item. Each resource requirement is composed of at least one *filtering condition* (what type of resource) and one *quantity expression* (how many resources). Filtering conditions are discussed on page 69 and quantity expressions are described on page 71.

Creation

Resource requirements are created:



- Manually through the Resource Requirements tab in the dialog of the Resource Manager block as discussed on page 75and described in the tutorial that starts on page 23.
- By default, as discussed on page 76 and shown in the tutorial that starts on page 26 (the default is 1 resource with a single resource property).
- Manually using an external application, such as an Excel worksheet, as described in the tutorial that starts on page 31.

Steps to take before creating

Before creating a resource requirement, it is helpful to perform the following steps:

1) Identify the components that will be needed:
   a)Determine how many distinct collections of resources are required, where each collection of resources is defined as being those resources sharing identical values for a specified set of properties
   b)Identify the properties and property values that are common to each resource collection
   c)Determine the quantity required from each collection
   d)Determine if alternative collections could be considered if the quantity cannot be obtained from a particular collection
   e)Identify the criteria for any of those alternative collections
   f)Determine the quantity required from any of the alternative collections
2) Use the results of the process above to determine how many quantity expressions are needed. A quantity expression will be required for each distinct resource collection from which a quantity will be selected.
3) Once the quantity expressions have been identified, determine how many different filtering conditions are needed to represent all of the distinct resource collections referred to in step 1. Filtering conditions are used in quantity expressions to generate the collection of resources from which the specified quantity is selected.

☞ See “Tutorial 3: Use Attributes for Flexibility and Scalability” for examples showing how to use item attributes to minimize the number of resource requirements a model needs.

Filtering conditions

Creating one or more filtering conditions is the first necessary step in defining a resource requirement. Filtering conditions are the smallest component of a resource requirement. The purpose of a filtering condition is to identify a collection of one or more resources that match a specific criterion. Filtering conditions are based on a relation between a particular resource or group property and values for that property. For example, a filtering condition could be that the resource comes from a particular pool or that it costs less per use than a specific amount.

Saved filtering conditions are used in quantity expressions, discussed on page 71. When a quantity expression gets executed, the resources it can use is limited to those that meet the criteria of the quantity expression's filtering conditions.

Types of filtering conditions

Each filtering condition filters on one, and only one, property of either a resource or group member:

1) A *resource filtering condition* filters on one specific resource property. All pre-defined and custom resource properties are listed in the Property popup menu shown in the screenshot to the right. Examples of resource properties are: pool, total idle time, or a group. (See "Resource properties" on page 61.)

2) A *group filtering condition* filters on one specified group property – either Skill Level or Resource Rank. (See "Group properties" on page 66.) There must be at least one group of resources in the model to enable group filtering conditions.

Note that a group filtering condition is associated with a group property—either Skill Level or Resource Rank—and not with a particular group. As discussed in "Using group filtering conditions" on page 74, a group filtering condition must be used in conjunction with a resource filtering condition that limits the collection of resources to a specific group.

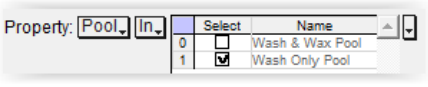
Creating a filtering condition

One way to create a filtering condition is by using the *Create/modify filtering conditions* frame in the Resource Manager's Requirements tab, shown here. Another method is to auto-generate the filtering condition either by defining it directly in the Advanced Resource database's Resource Requirement Expressions table or by using Excel or some other external application. Both approaches are shown in the Tutorial 1 chapter. In addition, filtering conditions can be automatically created by default when pools or groups are created, as discussed below.

☞ Each filtering condition filters for one condition of one property.

Conditional operators

Each filtering condition is composed of one conditional operator applied to one property. The possible conditional operators are: =, >, <, >=, <=, != (not equal), *BitAnd* (a bitwise AND comparison discussed in the Technical Reference),



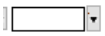
and *In* (for selecting the value of the property from a list of values, as shown here). As discussed above, the property can be a resource property or a group property.

Property value

To the right of the conditional operator popup is a field or table, depending on the property that has been selected and what is chosen in the property value source popup, discussed below. This field or table is where the *property value* is selected or entered. For example, in the above screenshot there is a table listing the existing pools. In the section on "Types of filtering conditions" on page 69, the two screenshots have fields after the conditional operator popups for entering a number.

Property value source popup

Shown here, the single downward arrow button to the right of the property value field is the *property value source popup*. Use this to select the source for the property value.



The source options for string properties are:

- Select from a list. Use this option to specify a fixed value for the property from a list of the existing values for the selected property. When this option is selected, the popup to its left will contain the list of values to choose from. Select one of the values in that popup.
- Use an attribute. Use this option to get the property value from an item's attribute. When this option is selected, the popup to the left will contain a list of the model attributes plus options to create new attributes. Select an existing attribute from this list or create a new attribute to get the property value from.

The source options for numeric properties are:

- Type a number. Use this option to specify a fixed numeric quantity for the property. When this option is selected, a parameter box appears to the left of the property value source popup. Input the value you want the property to be equal to whenever the filtering condition is executed.
- Use an attribute. Use this option to get the property value from an item's attribute. When this option is selected, the popup to the left will contain a list of the model attributes plus options to create new attributes. Select an existing attribute from this list or create a new attribute to get the property value from.
- Use a system variable. When this option is selected, the popup to the left lists the ExtendSim system variables such as currentScenario, currentSim, and currentTime, For a complete list of system variables, see the Technical Reference.

Default filtering conditions and resource requirements

When a Resource Pool block is used to create a pool, one filtering condition (as part of a resource requirement) is automatically created for each pool. If a Resource Manager block is used to create a pool or group, automatically creating a default filtering condition is optional. Furthermore, whenever a new filtering condition is created there is an option to automatically create a default resource requirement of 1 filtered resource. Default filtering conditions have "FC_" before the name of the pool or group. See "Default resource requirements" on page 76.

Testing

Once the property and condition selections have been made, use the Resource Manager's Test button to apply the selections to the model's resources. This opens a table displaying all the

resources that match the selected filtering condition. For a group filtering condition, every resource that has the specified group property will be displayed, no matter which group it is in.

For the definition of a filtering condition, select it in the *Choose filtering condition* popup menu of the Resource Manager's Requirements tab. The settings will be for the selected filtering condition.

Saving
A new or modified filtering condition won't be available for use until the Save or Save As button has been clicked. Until the filtering condition is saved, or while it is being modified, its name will be displayed in red.

☞ Saving the filtering condition saves it to the database. To save the database, save the model.

Deleting a filtering condition
If you try to delete a filtering condition, it may give an error message that it is being used by a quantity expression. Before the filtering condition can be deleted, any quantity expressions that use it must also be deleted. And before those quantity expressions can be deleted, any resource requirements that use them must also be deleted.

Examples of filtering conditions
Examples of filtering conditions include:

- FC#1 – filter using a *resource* property *Pool* that has the condition = *Pool 2*
- FC#2 – filter using a *resource* property *Total Idle Time* that has the condition > *10*
- FC#3 – filter using a *group* property *Skill Level* that has the condition = *2*
- FC#4 – filter using a *resource* property *Group* that has the condition = *Group A*

As discussed below, saved filtering conditions are used to create a quantity expression.

Quantity expressions

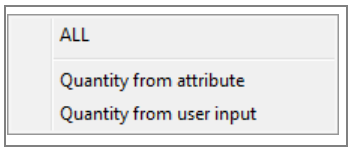
Defining one or more quantity expressions is the second step towards creating a resource requirement. A quantity expression is a logical statement that is used to select a certain quantity of resources from an identified collection of resources. The collection of resources is limited by all the filtering conditions that are entered in the expression.

The expression can be quite complicated. However, for each quantity expression only the specified quantity of qualifying resources will be allocated to an item.

The three clauses of a quantity expression

Each quantity expression consists of three clauses:

1) A *Quantity* clause that specifies how many resources to select from a resource collection. The quantity is entered using the Quantity popup menu at the right of the SELECT button. As seen in the screen shot to the right, the choices for the popup menu are:



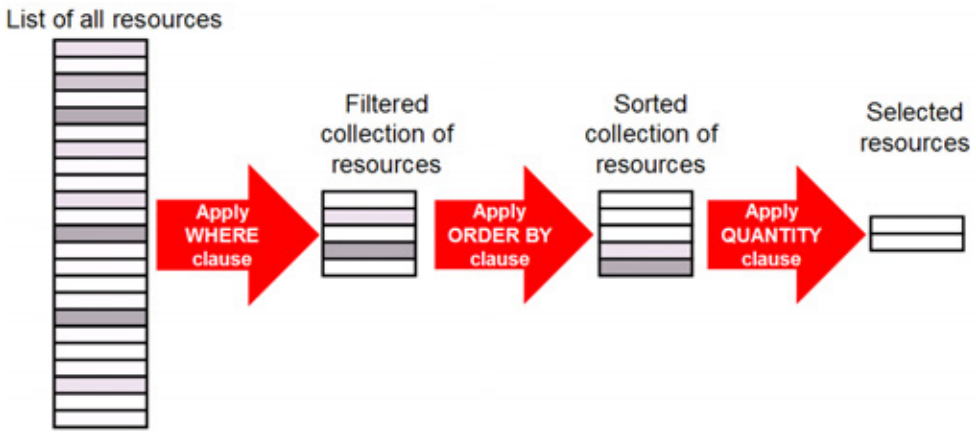
- All the resources
- A quantity determined from a named attribute
- A quantity that is user-input (the default)

Unless it is limited by a WHERE clause in the quantity expression, the selected number of resources will be drawn from all the resources in the Resources table of the Advanced Resources database.

2) A *WHERE* clause that contains the criteria that resources must satisfy in order to be selected. The WHERE clause generates the resource collection; the criteria are specified using one or more filtering conditions.

3) An optional *Order By* clause that is used to sort the collection of resources generated by the WHERE clause. If chosen, any one of the resource properties listed on page 61 can be used to sort the resources. The collection of resources is sorted based on the values (in ascending or descending order) of a selected resource property. The resources at the top of the sorted list will be used first.The default is that an ascending value of utilization is used to sort the resources. This causes resources that are being utilized the least to be selected first.

When the simulation is run, the quantity expression will be applied as shown below:

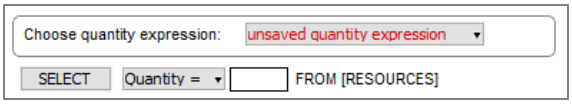


How a quantity expression gets applied during the simulation run (“Order By” is optional)

How to create a quantity expression

There are several ways to create a quantity expression:

1) One method is to use the *Create/ modify quantity expressions* frame in the Resource Manager’s Requirements tab.



2) Another method is to auto-generate the quantity expression either by defining it directly in a table in the Advanced Resource database or by using Excel or some other external application.

☞ The above methods are shown in the Tutorial chapters.

3) In addition, quantity expressions can automatically be created by default when pools or groups are created, as discussed below.

No matter how created, each quantity expression specifies the quantity of resources to be drawn from a set of resources defined by filtering conditions. The result is then saved as a quantity expression for use when building a resource requirement.

☞ When a new quantity expression is created, a corresponding resource requirement is optionally created. See "Default resource requirements" on page 76.

Default quantity expressions

When the *Resource Pool* block is used to create a pool, a single quantity expression with a quantity of 1 is automatically created. If instead the Resource Manager block is used to create a pool or group, default quantity expressions are optional.

☞ After a quantity expression has been created, the quantity can be changed to any of the three choices shown below, as shown in "Change the quantity to 2 Washer/Waxers" on page 26.

Creating a quantity expression

To create a quantity expression using the Resource Manager block's Requirements tab, check the *Create/modify quantity expressions* frame to enable it. Then either choose an existing quantity expression from the popup menu and modify it or create a new quantity expression.

To create a new quantity expression:

- First determine if the expression should start with all the resources from the Resources table or only a specified number of those resources—see the Quantity clause on page 71. (Optionally, you can also decide if the resources should be sorted by a property; see the Order By clause on page 72). Resources at the top of the list will be chosen first.
- Then click the *SELECT* button to place the quantity in the expression area.

At this point, the quantity expression requires a specified quantity (all or a number) of resources. However, the collection of resources is still *all* the resources in the Resources table of the Advanced Resources database.

☞ In models with large numbers of resources, it will speed simulation run times if the first part of the expression uses a filtering condition that limits the search area to a pool or group.

To narrow the search area, click the WHERE operator. Then insert a filtering condition; typically this would be a resource filtering condition that references a Pool or Group (for example, *Property: Pool = Pool 2*).

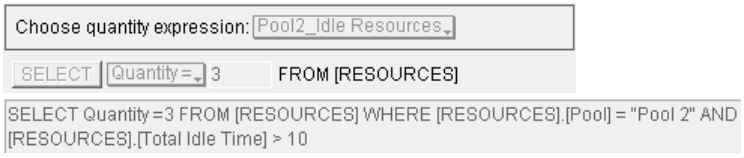
To use additional filtering conditions, use the AND/OR operators in conjunction with grouping parentheses. The result is displayed in the expression area.

☞ See below if group filtering conditions are used.

Example

The quantity expression named *Pool2_IdleResources* is shown in the screen shot below. It was created using a quantity of 3 and two of the filtering conditions (FC#1 and FC#2) that were shown in "Examples of filtering conditions" on page 71.

The quantity expression states that 3 resources will be required and that each resource must come



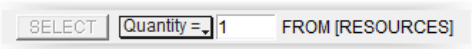
from Pool 2 (FC#1 is True) and have a total idle time > 10 (FC#2 is True).

Changing the quantity in a quantity expression

The ARM system allows for changing the quantity in the quantity expression portion of a resource requirement. This is particularly useful when:

- You have allowed the system to create a default resource requirement, which by definition requires 1 resource, but you want a higher quantity (as in this tutorial).
- You want the default resource requirement to get its quantity from the attributes on items.

When creating a quantity expression, the "Quantity =" popup menu gives three options:



- All filtered resources
- Quantity from attribute
- Quantity from user input

By default, the expression will use any number you enter in the field at the right, as in this example. However, it is more common that you would use an attribute value to specify a quantity.

Using group filtering conditions

When creating a quantity expression that references a group, group filtering conditions must be used in conjunction with a resource filtering condition that specifies a particular group. This is accomplished by choosing Group as the property for the resource filtering condition.

For example, assume the members of Group A and Group B all have Skill Level = 2. If the quantity expression includes a condition that filters on the *group* property (Skill Level = 2), it must also include a second condition that filters on a *resource* property that is the Group (either *Group = Group A* or *Group = Group B*).

☞ When creating a quantity expression, group filtering conditions must be used in conjunction with a resource filtering condition that specifies a particular group. And the resource filtering condition must be specified before the group filtering condition can be added to the expression.

Testing

The application of a quantity expression selects a specified quantity of specified resources. It is then used to create the resource requirement. To determine if the expression selects the resource that is wanted, use the Test button.

Saving

A new or modified quantity expression won't be available for use until the Save or Save As button has been clicked. Until the quantity expression is saved, its name will be displayed in red.

☞ Saving the quantity expression saves it to a database. To save the database, save the model.

Deleting a quantity expression

If you try to delete a quantity expression, it will probably give an error message that the quantity expression is being used by a resource requirement. Before the quantity expression can be deleted, any resource requirements that use it must also be deleted.

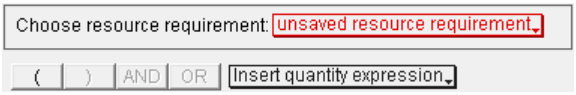
Resource requirement creation

The final step is to create the actual resource requirement. Resource requirements are logical expressions consisting of one or more quantity expressions combined using AND/OR operators in conjunction with grouping parentheses. They describe the resources needed by an item in order for it to perform a task in the model.

☞ See also "Resource requirement overview" on page 67.

Methods for creating a resource requirement

Resources requirements can be created:



- Manually through the Requirements tab in the dialog of the Resource Manager block as discussed below and described in the tutorial that starts on page 23.
- By default, as discussed on page 76 and described in the tutorial that starts on page 26 (the default is 1 resource with a single resource property).
- Manually using an external application, such as an Excel worksheet, as shown in the tutorial that starts on page 31.

Resource requirement rank

When it is created, each resource requirement gets a *resource requirement rank*. The resource requirement rank must be ≥ 0; the default rank is 1. This ranking is used for sorting items that are waiting for resources, as discussed in "Resource allocation policy" on page 78. It can be overridden for a particular resource, as discussed in "Overriding the resource requirement rank" on page 80.

Creating a resource requirement manually

On the Requirements tab, enable the *Create/modify resource requirements* frame. Then either choose an existing resource requirement from the popup menu and modify it or create a new one.

To create a resource requirement, either choose a quantity expression from the popup menu or start with a grouping operator. Use the AND and OR operators between quantity expressions; use the grouping operators to group quantity expressions. The resource requirement rank can be changed from the default of 1.

Testing

To determine if the expression selects the resources that are wanted, use the Test button.

Saving

A new or modified resource requirement won't be available for use until the Save or Save As button has been clicked. Until the resource requirement is saved, its name will be displayed in red.

☞ Saving the resource requirement saves it to the Advanced Resource database. To save the database, save the model.

Default resource requirements

Under certain circumstances the ARM system creates a default resource requirement. In some cases this is done automatically and in other cases it is optional.

The circumstances when a default resource requirement could be created are:

- Every time a new pool is created using a Resource Pool block
- Optionally when a new pool is created using a Resource Manager block
- Optionally when a new group is created using a Resource Manager block
- Optionally when a new quantity expression is created using a Resource Manager block
- Optionally when a new filtering condition using a Pool is created using a Resource Manager block

Created by default when a new pool or group is created

When a default resource requirement is created for a new pool or group, 3 components are created:

1) A filtering condition named “FC_[name of pool or group]”, with the condition that:
   - If a new pool is created, the resource must come from the new pool (“Filter using properties of resources, where the property = [the new pool]”).
   - If a new group is created, the resource must come from the new group (“Filter using properties of resources, where the property = [the new group]”).
2) A quantity expression named “1 from [name of pool or group]”, indicating that the requirement is for exactly 1 resource from the pool or group.
3) A resource requirement named “RR_[name of pool or group]”. (When a resource requirement is automatically created for a new pool or group, it is the same as the quantity expression.)

Because the default filtering condition is the newly created pool or group and the default quantity expression is 1 resource from that pool or group, the default resource requirement is that 1 resource is required from the new pool or group.

☞ After a resource requirement has been created, the quantity in its quantity expression can be changed as shown in “Change the quantity to 2 Washer/Waxers” on page 26.

Created when a filtering condition or quantity expression is created

If a default resource requirement has not already been created through one of the above mechanisms, the option will be given to create a corresponding default resource requirement when a new filtering condition or quantity expression is created. The default resource requirement will be the same as the filtering condition or quantity expression, but its name would be preceded by “RR”. The name of the resource requirement can be changed before the requirement is saved.

For example, when creating a new filtering condition the ARM system would ask if you want to create a default resource requirement that is the same as that filtering condition with a quantity of 1, such as 1 resource from the Wash Only pool. The quantity can be changed as discussed in “Change the quantity to 2 Washer/Waxers” on page 26.

Using an external application to create a resource requirement

Resource requirements can be defined in external applications such as Excel, SQL Server, and Access and imported into the model's Advanced Resources database using the Data Import Export block (Value library). The requirements are defined using the structure of the Resource Requirement Expressions database table and adding content. After the definitions have been imported, ARM *creates* the resource requirements either the first time the simulation is run or when the resources are auto-generated.

This process is described in detail in "Tutorial 2: Resource Creation Part 2" on page 31.

Release rules

Release rules determine if and when resources are released from the items they've been allocated to. Custom release rules are created in the Resource Manager block. Each Resource Pool Release block can release resources using those custom rules or using other criteria set in its dialog.

☞ The only way to release resources that have been allocated to an item is by passing the item through a Resource Pool Release block (Item library).

Release Rules tab of Resource Manager block

This tab is for creating and modifying *custom resource release rules*. Custom release rules provide a user-definable mechanism for controlling which resources to release. These rules use the same mechanism as resource allocation (the execution of a resource requirement), to select from a collection of filtered resources. However, when a resource requirement is used by an item entering or residing in a Queue, ARM uses the requirement to select viable resources to allocate to the item from the list of resources in the Resources database table. Whereas when a resource requirement is used in a custom release rule, ARM uses the requirement to select viable resources to release from the item from the list of resources already allocated to the item.

Custom resource release rules can be based on any resource requirement and can be used by any Resource Pool Release block anywhere in a model.

To create or modify a release rule:

- Go to the Resource Manager's Release Rules tab
- Choose an existing resource release rule or create a new one
- Select the resource requirement that the rule should be associated with
- In the table, specify the Release Quantity by selecting from the popup menu—All, None, a Value, or Get from an Attribute

Resource Pool Release block

When its release behavior is set to *Release: advanced resources*, the dialog of the Resource Pool Release block gives three choices for controlling which resources to release when items enter the block. The options are to release:

- All resources
- All resources from a specific resource order
- Resources using a custom rule

☞ Custom rules are created in the Release Rules tab of the Resource Manager block, discussed on page 77.

Exclusion rules

The dialog of the Resource Pool Release block also provides customizable *exclusion rules* that give an additional level of control over which resources get released from passing items. Exclusion rules allow modelers to create exceptions to any of the above three resource release options and prevent specified resources from being released from passing items.

Note that, if an item has not had all of its resource removed before being destroyed, e.g., entering an Exit block, the residual resources will remain in the busy state for the remainder of the simulation run and will not be available for reuse. While this could be done intentionally to represent consumable resources, in general it would be a modeling error that would produce incorrect results.

Policies

The Policies tab of the Resource Manager block is for:

- Setting a policy that determines the order in which items are evaluated for resource allocation, as discussed below.
- Enabling allocated resources to be reassigned to higher priority resource orders.
- Overriding the resource requirement rank for individual resources, as discussed on page 80.

Resource allocation policy

The settings in the top section of the Policies tab control the order in which idle resources search for waiting items before attempting allocation. This determines which items a resource can allocate itself to when the resource becomes available for work.

An opportunity for resource allocation occurs when either of the following events occur:

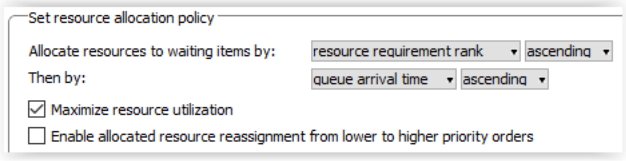
4) An item arrives in an ARM-mode queue block and selects resources—*push* events. The resource allocation policy is not used.
5) A resource transitions to being available for allocation and selects an item—*pull* events. The resource allocation policy is used.

Push events

Resource allocations that result from item arrivals in ARM queues are referred to as “push” allocations because the item is pushing itself to idle resources. Because there can be more resources available than are required by an item, the *item* must decide which of the available resources to select. This decision is made using the logic embedded in the associated resource requirement. Consequently, push event-based allocations do not use the resource allocation policy.

Pull events

Resource allocations that result from resources transitioning to idle are referred to as “pull” allocations because items waiting for resources are being *pulled* to available resources. When idle resources pull waiting items,



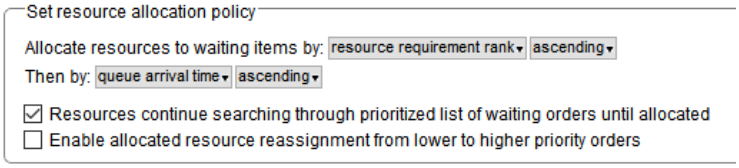
they use the resource allocation policy to determine the order in which to search for waiting items and when to terminate their search. The pulling resource searches the sorted list of waiting items until one of the following occurs:

1) Either an item is encountered that can have its requirement satisfied by using the pulling resource
2) Or the search termination condition is encountered

Creating an allocation policy

The *Set resource allocation policy* frame in the Resource Manager's Policies tab contains the controls for setting the resource allocation policy for "pull" allocations. The popups in this frame determine the order in which idle resources search for waiting items to allocate themselves to.

Sorting criteria



At any time during a simulation run, items waiting to have their resource requirements satisfied are sorted into a prioritized list. The sorting is accomplished using item information and the settings in the "Set resource allocation policy" frame of the Resource Manager's Policy tab:

- The primary sorting key is: *Allocate resources to waiting items by...*
- The secondary sorting key is: *Then by*

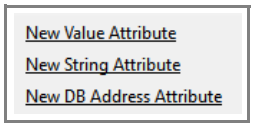
These options tell ARM what item information to use when sorting the list of waiting items.

☞ The default is to sort items by the ranking of their resource requirements and then by the time the item arrived to the queue.

Primary sorting key popup

The primary sorting uses one of the following criteria about the items:

- **The resource requirement rank**. As discussed on page 75, the resource requirement rank is a property of the resource requirement. It is specified in the "Create/modify resource requirements" frame of the Requirements tab; the default is 1. While a model is running, each outstanding (unsatisfied) request for resources is associated with both a resource requirement and a waiting item. Thus, each item waiting for resources has an associated resource requirement rank that can be used for sorting.
- **Item attributes**. Any Value or DB Address attribute that is used for sorting must resolve to a readable cell in any one of the model databases.
- **The item priority**.
- **The item's queue arrival time**. The queue arrival time is recorded in the resource order record associated with a waiting item.



The popup next to the primary sorting key is for specifying whether the selected information is to be sorted in *ascending* or *descending* order.

Secondary sorting key popup
The secondary sorting key popup is only made available when the "Resource requirement rank" option is selected as the primary sorting key. This secondary popup has all of the options the primary sorting key popup has, except for the resource requirement rank option. The secondary key can be sorted by ascending or descending order.

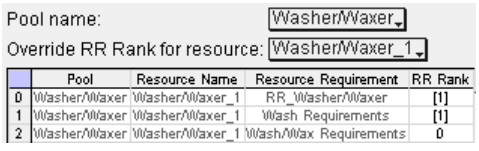
The checkboxes
By default, the "Resources continue searching..." checkbox is checked. This is used to control when the search for waiting items is terminated.

- When it is checked, the Resource Manager searches all waiting items that the pulling resource can service (be allocated to) until either one of the following is true:
  - The resource is allocated to a waiting item
  - Or, the end of the list of waiting items is encountered
- When this checkbox is unchecked, the Resource Manager searches all waiting items that the pulling resource can service (be allocated to) until:
  - The resource is allocated to a waiting item
  - An item is encountered for which the resource requirement can't be satisfied

Naturally, checking the this checkbox will increase resource utilization because ARM will capitalize on every opportunity to satisfy the requirement of a waiting resource. However, running in this mode will cause resources to be allocated to lower priority resource orders. Either way, ARM will always search to satisfy waiting resource orders in the order specified by the settings of the popups in the "Set resource allocation policy" frame.

Overriding the resource requirement rank
As discussed in "Sorting criteria" on page 79, one of the properties that waiting items can be sorted by is the *resource requirement rank* (RRR). When the RRR is specified as being part of the resource allocation policy, the rank value of each waiting Resource Order's resource requirement is used to sort the list of waiting items before idle resources begin searching for a waiting item to allocate themselves to. A resource requirement rank must be ≥ 0; the default resource requirement rank is 1.

The original value of the RRR is entered on the Requirements tab when the resource requirement is created. This value can be overridden by individual resources such that, when searching for waiting items to allocate themselves to, the resources can use their overriding rank values to change how the list is sorted. This means that, only for the specified resource, a resource requirement's rank will differ from the original value entered on the Requirements tab. Since a resource requirement could be composed of multiple resources, each resource that is part of a resource requirement can override the resource requirement's rank with a different value.



Wash/Wax resource requirement is ranked higher

Overriding is accomplished by changing the values in the RR Rank column in the table labeled *Override resource requirement rankings*.

☞ The original ranking values are displayed in square brackets in the RR Rank column.

The Advanced Resources database

The Advanced Resources database provides the architectural framework for the ARM system. ARM uses the database to represent, manage, and track the status and properties of resources, pools, and groups, as well as resource allocation and release rules. The database is automatically created when a Resource Manager block is added to a model. It is easily accessed by clicking the View Database button in the Resource Manager's tabs or through the ExtendSim Database menu.

There are several tables in the database. The most relevant to the modeler are described below.

Table	Description
Resources	The repository for all the resources in a model. Each record represents a specific quantity of resources having a common set of input and output properties. This table is where resource status and thus, availability is managed. It is also where utilization and the history of time spent in each resource state is recorded.
Pools	A summary of the Resources table from a pools perspective. Each record contains a summary of information about all of the resources in a particular pool. Summarizes the results from the simulation run (total idle time, total utilization, and so forth) for each pool.
Resource Requirements	Provides queuing, arrival, and departure statistics for each resource requirement in the model.
Resource Classes	Provides a dialog-independent mechanism for defining all of a model's resources, pools, groups and custom resource properties in a single table. ARM uses this table to automatically instantiate these components in a model during model initialization. Because everything needed to fully characterize the resources, pools, groups, and custom resource properties can be specified in this one table, modelers can populate and import the contents of this table from an external application such as Excel, SQL Server, or Oracle.
Resource Requirement Expressions	Provides a dialog-independent mechanism to define all of a model's resource requirements, quantity expressions and filtering conditions in a single table. ARM uses this table to automatically instantiate these components in a model during model initialization. Because everything needed to fully characterize the resource requirements, quantity expressions, and filtering conditions can be specified in this one table, users can populate and import the contents of this table from an external application such as Excel, SQL Server, or Oracle.
Resource Transaction Log	Depending on the settings in the Resource Manager's Event Logging tab, this table displays the transactions, status transitions, and results of the simulation run.

Also see “Primary transactions for ARM” on page 83.

Blocks that compose the ARM system

The Resource Manager block is central to the ARM architecture and several other blocks perform important functions.

Block (Library)	Function When Used With Advanced Resources
Resource Manager (Item)	This block is the brain of the ARM system. It centralizes all of the resource information in the model and uses this information to make all resource allocation decisions. It provides an interface for managing advanced resources, resource requirements (allocation rules), release rules, and transaction logs. Use it to: • Create, edit, and delete resources, pools, groups, and their properties • Create resource requirements • Define release rules • Define policies for allocating idle resources to items • Enable event logging • View statistical results
Batch (Item)	Takes all of the ARM resources that are attached to inbound items and attaches them onto the outbound batch item.
Queue (Item)	Controls the ordering and allocation of resources Note: Selected queue behavior must be set to *advanced resource queue*.
Queue Equation (Item)	The equation can be used to conditionally allocate requirements to items released from the Queue Equation block. Note: Enable advanced resources (AR) on the block’s Options tab.
Query Equation (Item)	Can be used to share resources across multiple items when items are spawned with attached ARM resources.
Resource Pool (Item)	A shortcut for creating a pool of resources and a way to add or removeresources from Pools during or at the start of the simulation. Controls the initial properties and number of resources in a pool and reports aggregated results for its resources. Notes: 1) Selected behavior must be *advanced resource pool*. 2) Since all its capabilities can be achieved using the Resource Manager block, this block is optional. 3) Resource Pool resources do not have properties unless the resources are advanced.
Resource Pool Release (Item)	Controls the release of resources. Notes: 1) Selected behavior must be *release advanced resources*. 2) This block provides the only mechanism for releasing ARM resources from items.
Statistics (Report)	Reports statistics for individual resources. Especially useful for exporting data and for performing batch means and multi-run analyses.

Block (Library)	Function When Used With Advanced Resources
Shift (Item)	ARM resources can be associated with a Shift block's shift. Whenever the status of a shift changes, the associated Shift block notifies the Resource Manager block and the Resource Manager block updates the status values of the affected resources. See the *ARM Shift Changes using Scenario Manager* model located at examples/Discrete Event/Resources and Shifts/Advanced Resource Mgt.
Unbatch (Item)	Can be used to share resources across all of the unbatched items when the inbound items have ARM resources attached to them.

Primary transactions for ARM

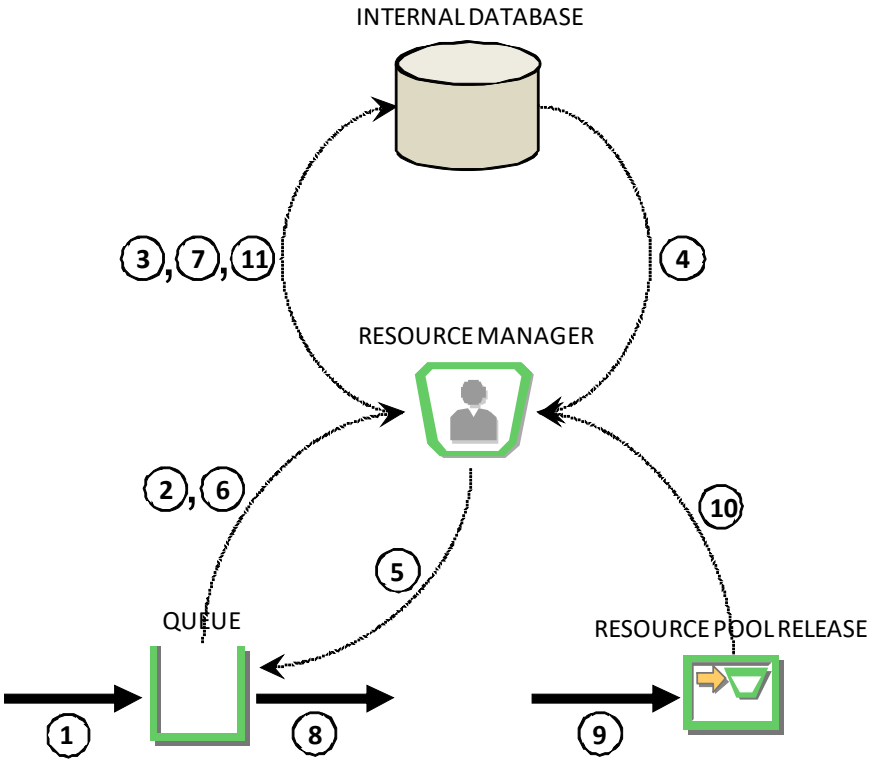
The following diagrams and tables give an overview of how advanced resources are used in a model. The explanation is divided into sections which describe what happens when:

- An item arrives at a Queue that is in ARM mode
- A Shift block used by the ARM system changes status
- A Resource Pool block in ARM mode gets a message to add or remove advanced resources
- An item passes through a Resource Pool Release block that is in ARM mode

☞ In the explanations, the Database Tables Used column has the following abbreviations: RO (Resource Orders), RTL (Resource Transaction Log), and RR (Resource Requirements).

Item-initiated ARM transactions

This part explains what happens when an item arrives at a Queue that is part of the ARM system.

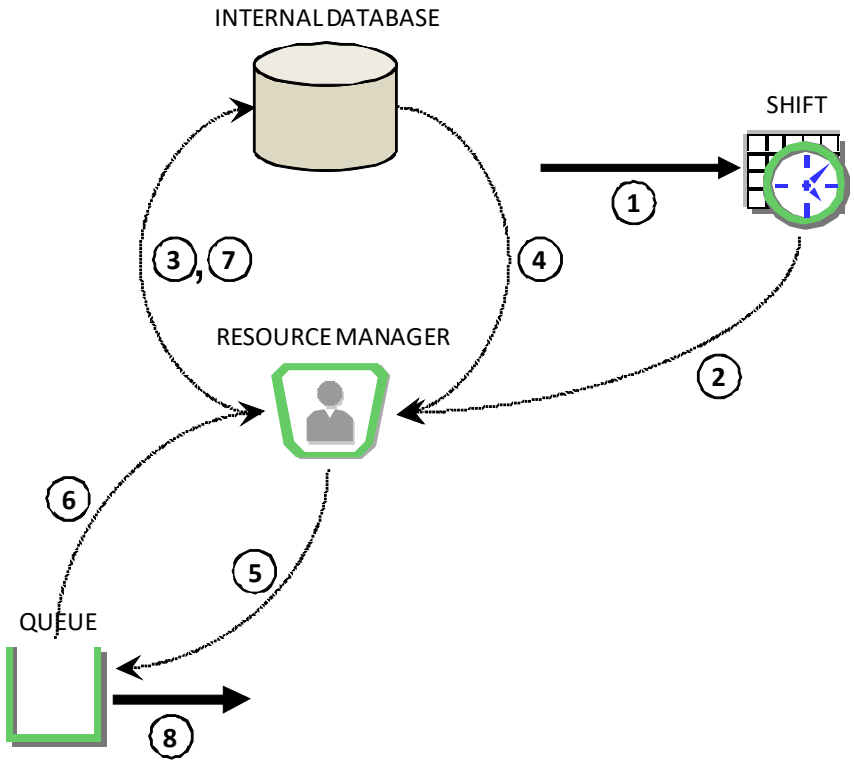


#	Description	Message-Sending Block	Message-Receiving Block	Database Tables Used
1	An advanced resource item arrives in the Queue.		Queue	
2	The Queue orders resources from the Resource Manager.	Queue	Resource Manager	
3	The Resource Manager generates a resource order and assigns a Resource Order ID in the database and to the item.			RO, RTL
4	The Resource Manager checks for availability of required resources.			RO, Resources, RR, RTL

#	Description	Message-Sending Block	Message-Receiving Block	Database Tables Used
5	The Resource Manager notifies the Queue when the required resources are available for the item.	Resource Manager	Queue	
6	The Queue checks if it is blocked. If it is not, it grants permission to the Resource Manager to allocate the resources to the item.	Queue	Resource Manager	RO, Resources, RTL
7	The Resource Manager allocates the resources and changes their status to *busy*.			RO, Resources, RTL
8	The item with the resources allocated to it leaves the Queue and is released to the model.	Queue		
9	The item arrives in the Resource Pool Release.		Resource Pool Release	
10	The Resource Pool Release sends a command to the Resource Manager to release resources from the item.	Resource Pool Release	Resource Manager	RO, Resources, Resource Release Rules, RTL
11	The Resource Manager releases resources from the item and changes the status of the released resources from *busy* to their new status (idle, off shift, etc.)			RO, Resources, RTL

Shift

When advanced resources use a Shift block, the transactions are as follows.

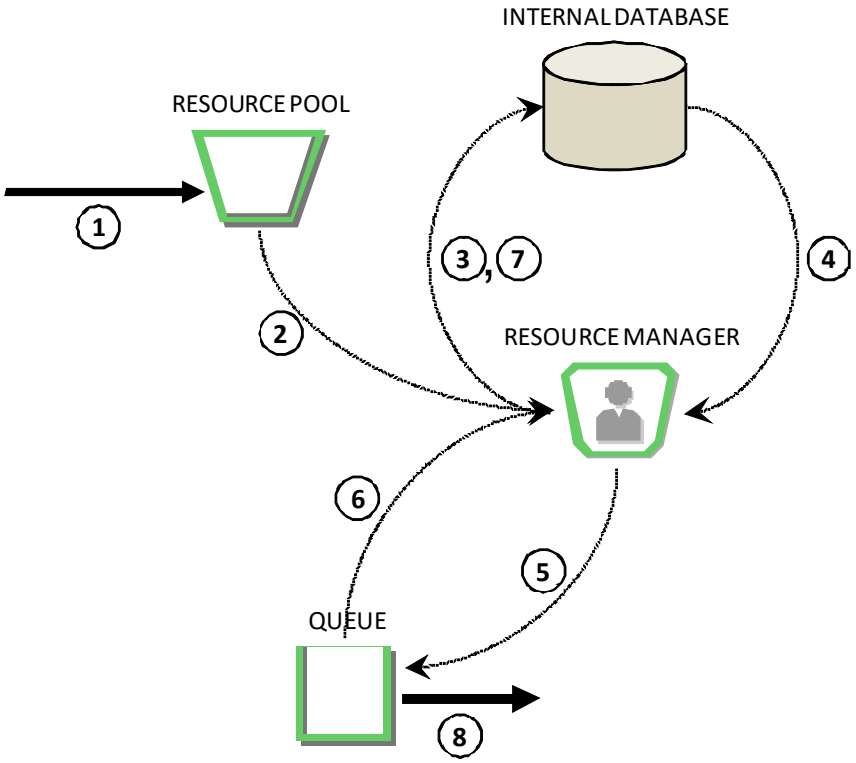


#	Description	Message-Sending Block	Message-Receiving Block	Database Tables Used
1	The Shift receives a status change message.		Shift	
2	The Shift sends a shift status change message to the Resource Manager.	Shift	Resource Manager	
3	The Resource Manager changes the status of all resources that use the Shift.			Resources, RTL
4	If any resources have transitioned to the *idle* status, the Resource Manager attempts to service waiting items that depend on those resources.			RO, Resources, RR, RTL

#	Description	Message-Sending Block	Message-Receiving Block	Database Tables Used
5	If items can be serviced, the Resource Manager notifies Queues with waiting items that the required resources are available.	Resource Manager	Queue	
6	The Queues check for being blocked. If they are not, they grant permission to the Resource Manager to allocate the idle resources to their items.	Queue	Resource Manager	RO, Resources, RTL
7	The Resource Manager allocates the resources and changes their status to *busy*.			RO, Resources, RTL
8	The Queues release the items.	Queue		

Resource Pool-initiated ARM transactions

When a Resource Pool block receives a message during the simulation run that resources need to be added or removed, the transactions occur as illustrated below.



#	Description	Message-Sending Block	Message-Receiving Block	Database Tables Used
1	The Resource Pool receives a message to add/remove resources.		Resource Pool	
2	The Resource Pool notifies the Resource Manager that the quantity of pool resources has changed.	Resource Pool	Resource Manager	
3	The Resource Manager changes the quantity of resources for the pool.			Resources, RTL
4	If resources were added, the Resource Manager attempts to service waiting items.			RO, Resources, RR, RTL

#	Description	Message-Sending Block	Message-Receiving Block	Database Tables Used
5	If items can be serviced, the Resource manager notifies Queues with waiting items that the required resources are available.	Resource Manager	Queue	
6	The Queues check for being blocked. If they are not blocked, they grant permission to the Resource Manager to allocate the idle resources to the items.	Queue	Resource Manager	RO, Resources, RTL
7	The Resource Manager allocates the resources and changes their status to *busy*.			RO, Resources, RTL
8	The Queues release the items.	Queue		

What's new in ARM since 9.0

ARM has been used for a few years now and many customers use this powerful tool to solve real world problems. We appreciate their feedback, which allowed us to add more capabilities to ARM since its original release.

☞ For additional and up-to-date information, see the Help of the Resource Manager block.

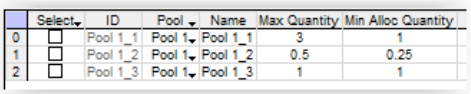
Fractional resources, resource classes, and multi-tasking

Prior to release 10, ARM required "**one record = one resource"**—each record in the Resources table represented a single resource. ARM now supports:

- *Fractional resources*, where each record can represent a fraction of a resource, so parts of one resource can be allocated to different items
- *Multiple resources*, where resources with an identical set of properties are represented in a single record in the Resources table (note that there is a reduction of fidelity with this option)

New fields in the Resources table support the fractional resources, resource classes, and multi-tasking features described above:

- *Maximum Quantity*: Specify how many resources are represented in the record.
- *Available Quantity*: A read-only field that reports the amount of Maximum Quantity currently available for allocation.
- *Minimum Allocation Quantity*; Define the smallest quantity of a resource that can be allocated to a resource order. Must be equal to or less than the Maximum Quantity.

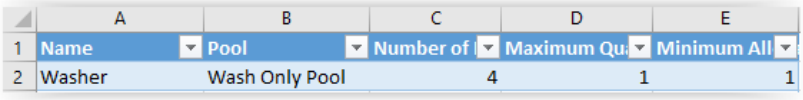


	Select	ID	Pool	Name	Max Quantity	Min Alloc Quantity
0	☐	Pool 1_1	Pool 1	Pool 1_1	3	1
1	☐	Pool 1_2	Pool 1	Pool 1_2	0.5	0.25
2	☐	Pool 1_3	Pool 1	Pool 1_3	1	1

See also "Fractional and multiple resources" on page 52.

External definitions and auto-generation

Modelers can now *define resources, pools, groups, custom resource properties, and resource requirements in Excel, SQL Server, Access*, and so forth, then use the Data Import Export block (Value library) to import those definitions to the Advanced Resources database. A pre-defined Excel template is included with ExtendSim. Any or all of the fields can be defined in that workbook for importing to the Resource Classes and Resource Requirement Expressions tables in the Advanced Resources database.

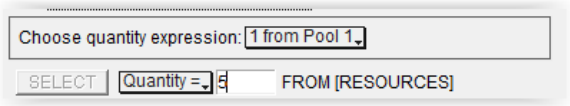


	A	B	C	D	E
1	Name	Pool	Number of	Maximum Qu	Minimum All
2	Washer	Wash Only Pool	4	1	1

When resources and requirements have been externally defined, the Resource Manager's *Auto-Generate* button causes the imported information to automatically create ARM components. This consists of pools, groups, resources, and custom properties from the Resource Classes table and the filtering conditions, quantity expressions, and resource requirements from the Resource Requirement Expressions table.

Ability to edit quantity in an existing expression

The *quantity of an existing quantity expression can now be edited* in the Quantity Expression frame of the Resource Requirements tab.



ARM enhancements to other blocks

- The *Resource Manager block now coordinates with the Data Import Export block to control when data gets imported into Advanced Resources database tables*. The Resource Manager automatically triggers the imports of data for all Data Import Export blocks that are set to import into the ARM database tables. These imports are triggered before the simulation runs during PreCheckData to ensure that ARM database data is current before the Resource Manager block performs its initialization. This coordination is necessary, in part, to enable the new auto-generation feature discussed above.
- Enhancements to the Activity block allow items transitioning into certain states (off-shift, scheduled or unscheduled down, and reassignments) to be preempted or shutdown. This allows modelers to *suspend progress towards the servicing of resource orders when the allocated resources become unavailable*.
- Batch and Queue Matching blocks now deal better with *batched items that have associated ARM resources*. When those items are batched, a new AR Order ID is generated for each outgoing item. All of the ARM resources associated with the batched items are transferred to the new resource order and the items' previous resource orders are destroyed. Preserve Uniqueness must be off for this feature.

- Each item that has an ARM resource allocated to it is assigned a unique resource order ID property named AR Order ID. Modelers now have *direct access to the AR Order ID* using Equation, Equation(I), Read(I), Write(I), Get, Set, Queue Equation, Query Equation(I), and History blocks. The property is listed as "_AR Order ID" under the item properties section of all attribute popup lists.
- The Resource Pool Release block has been enhanced to provide that *specific resources can be excluded from being released*. The exclusion is specified by designating a resource, a resource requirement, or a resource status value.

Resource list queries

Modelers can communicate directly with the Resource Manager block from ModL code to invoke queries to *identify all the resources that are blocking a specified resource requirement* from being satisfied. The query is invoked by sending messages to the Resource Manager's BlockReceive7 message handler with the following arguments:

- SysGlobalInt30 = resourceRequirementRecord
- SysGlobalInt31 = itemIndex
- SysGlobalInt33 = AR_GetBusyRequirementResources_ActionType

Parent-child relations for custom resource properties

Custom properties can now have parent-child relationships. When a custom property is defined, a dialog appears asking whether that property is a child of a parent field. If Yes, a second dialog appears for selecting a parent table and field from within the Advanced Resources database. This relating of a property to a field in a table is good database practice as it ensures the data is normalized, i.e., defined in one place only.

Resource sharing

Resource sharing is a new feature that allows individual resources to be shared across multiple items at the same time. See page 54 for more information.

Resource recycling

Previously, only idle resources could be used to satisfy an item's resource requirement. *Recycling* is a new default feature that allows modelers to satisfy an item's resource requirement using busy resources that have already been allocated to an item. See "Resource recycling" on page 53 for more information.

Resource reassignment

Resource reassignment allows to ARM to reassign allocated resources from lower priority items that are blocking higher priority items from having their requirements satisfied. See page 55 for more information.

Index

Symbols

A-C

D-F

G-I

M-N

O-P

Q-S

T-V