# SimEvents®

## Getting Started Guide

# MATLAB®&SIMULINK®

MathWorks®

# How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

# Contents

# Key Concepts in SimEvents Software

**3**

<div style="text-align: right">

# Inspect Statistics

</div>

**4**

<div style="text-align: right">

# Create Discrete Event Systems Using MATLAB and Stateflow Software

</div>

**5**

<div style="text-align: right">

# Simulate Multidomain Models

</div>

**6**

**7**

# 1

# Introduction

# SimEvents Product Description

**Model and simulate discrete-event systems**

SimEvents provides a discrete-event simulation engine and component library for analyzing event-driven system models and optimizing performance characteristics such as latency, throughput, and packet loss. Queues, servers, switches, and other predefined blocks enable you to model routing, processing delays, and prioritization for scheduling and communication.

With SimEvents, you can study the effects of task timing and resource usage on the performance of distributed control systems, software and hardware architectures, and communication networks. You can also conduct operational research for decisions related to forecasting, capacity planning, and supply-chain management.

## Key Features

- Discrete-event simulation engine for multidomain system models
- Entities with custom data attributes representing tasks, packets, and items
- Blocks for queuing, service, routing, resource management, multicasting, replication, and batching
- Statistics generation for delay, throughput, average queue length, and other metrics
- Library authoring with MATLAB® or Stateflow® for custom schedulers, hardware and software constructs, and communication channels
- Block diagram animation and inspection for visualizing model operation and debugging
- Custom animation creation for monitoring entities and events

# Discrete-Event Simulation in Simulink Models

SimEvents software incorporates discrete-event system modeling into the Simulink time-based framework, which is suited for modeling continuous-time and periodic discrete-time systems. In time-based systems, state updates occur synchronously with time. By contrast, in discrete-event systems, state transitions depend on asynchronous discrete incidents called events. Some examples illustrate these differences:

- Suppose that you are interested in how long the average airplane waits in a queue for its turn to use an airport runway. However, you are not interested in the details of how an airplane moves once it takes off. You can use discrete-event simulation in which the relevant events include:

  - The approach of a new airplane to the runway
  - The clearance for takeoff of an airplane in the queue

- Suppose that you are interested in the trajectory of an airplane as it takes off. You would probably use time-based simulation because finding the trajectory involves solving differential equations.

- Suppose that you are interested in how long the airplanes wait in the queue. Suppose that you also want to model the takeoff in some detail instead of using a statistical distribution during runway usage. You can use a combination of time-based simulation and discrete-event simulation, where:

  - The time-based aspect controls details of the takeoff
  - The discrete-event aspect controls the queuing behavior

In a Simulink model, you typically construct a discrete-event system by adding various blocks, such as generators, queues, and servers, from the SimEvents block library. These blocks are suitable for producing and processing entities, which are abstractions of discrete items of interest. Examples of entities are packets within a communication network, planes on a runway, or trains within a signaling system. Asynchronous events that correspond to motion and changes in entity attributes through the system model update the states of the underlying system. Examples of states are lengths of queues or service time for an entity in a server.

One or more discrete-event systems can coexist with time-based systems in a Simulink model. This coexistence facilitates the simulation of sophisticated hybrid systems. You can pass signals from time-based components/systems to and from discrete-event components/systems modeled with SimEvents blocks. The combination of time- and event-based modeling facilitates the simulation of large-scale systems that incorporate

smaller subsystems from multiple environments. An example of a large-scale system might have physical modeling for continuous-time systems, such as electrical systems, which communicate via a channel modeled as a discrete-event system. A Simulink model can also contain a purely discrete-event system with no time-based components when modeling event-based processes. These systems are common in models that represent logistic and manufacturing systems.

## See Also

### Related Examples
· "Simulate a Hybrid System" on page 6-2

### More About
· "SimEvents Product Description" on page 1-2

### External Websites
· Tech Talks: Understanding Discrete-Event Simulation

# Related Products

| In this section... |
| --- |
| "Information About Related Products" on page 1-5 |
| "Limitations on Usage with Related Products" on page 1-5 |

## Information About Related Products

See Related Products (`http://www.mathworks.com/products/simevents/related.html`)..

## Limitations on Usage with Related Products

### Code Generation

SimEvents blocks do not support code generation using the Simulink Coder™ product in version 5.0 (R2016a). Before version 3.1.2 (R2010a), SimEvents blocks offered limited code generation support for rapid simulation. Since version 4.0 (R2011b), SimEvents blocks do not support code generation using the Simulink Coder product. Support for rapid simulation was removed because the improvements in normal model simulation performance for SimEvents models matched or surpassed the performance of rapid simulation in releases before version 4.0.

### Simulation Modes

SimEvents blocks do not support simulation using the Rapid Accelerator, Accelerator, Processor-in-the-Loop (PIL), or External mode.

### Model Reference

SimEvents blocks cannot be in a model that you reference through the Model block.

# See Also

## Related Examples

· "Simulate a Hybrid System" on page 6-2

## More About

# What Is an Entity?

Discrete-event simulations typically involve discrete items of interest. By definition, these items are called entities in SimEvents software. Entities can pass through a network of queues, servers, gates, and switches during a simulation. Entities can carry data, known in SimEvents software as attributes.

In a SimEvents model, where there are SimEvents blocks and Simulink blocks, the badge

 denotes the transition between time-based and event-based behavior. (A badge is an icon the software uses to flag issues or transitions.)

These storage blocks hold entities as they move through a model:

- Entity generators
- Queues
- Servers
- Terminators

**Note** Entities are not the same as events. Events are instantaneous discrete incidents that change a state variable, an output, and/or the occurrence of other events. See "What Is an Event?" on page 1-9 for details.

The table shows examples of entities in sample applications.

| Context of Sample Application | Entities |
|---|---|
| Airport with a queue for runway access | Airplanes waiting for access to runway |
| Communication network | Packets, frames, or messages to transmit |
| Bank of elevators | People traveling in elevators |
| Conveyor belt for assembling parts | Parts to assemble |
| Computer operating system | Computational tasks or jobs |

A graphical block can represent a component that processes entities, but entities themselves do not have a graphical representation. When you design and analyze your discrete-event simulation, you can choose to focus on:

- The entities themselves. For example, what is the average waiting time for a series of entities entering a queue?

- The processes that entities undergo. For example, which step in a multistep process (that entities undergo) is most susceptible to failure?

---

**Note** SimEvents entities are fundamentally the same as Stateflow messages.

---

# See Also

Entity Generator

## Related Examples
- "Inspect Structures of Entities"

## More About
- "What Is an Event?" on page 1-9
- "Role of Entities in SimEvents Models" on page 3-2

# What Is an Event?

| In this section... |
| --- |
| "Overview of Events" on page 1-9 |
| "Viewing Events" on page 1-9 |
| "Actions for Events" on page 1-10 |
| "Event Actions Assistant for Events" on page 1-10 |

## Overview of Events

In a discrete-event simulation, an event is an observation of an instantaneous incident that may change a state variable, an output, and/or the occurrence of other events. Events can correspond to changes in the state of an entity.

### Typical Event Sequences

Specify event actions based on entity status. A typical event sequence in a SimEvents model is:

1 The generation of an entity.

2 The advancement of an entity from an Entity Generator block to an Entity Server block.

3 The completion of service on an entity in a server.

4 The exit of an entity from one Entity Server block to an Entity Terminator block.

5 The destruction of an entity.

## Viewing Events

Events do not have a graphical representation. However, you can associate actions with events as described in "Actions for Events" on page 1-10. The SimEvents software maintains an event calendar with which you can interact using `simevents.SimulationObserver` methods. You can create a custom event observer using this class and its methods. For more information, see "Interface for Custom Visualization".

## Actions for Events

SimEvents lets you create custom actions to happen when an event occurs for an entity. Every event can have a corresponding action. You can write actions for many events using MATLAB code or Simulink Functions.

## Event Actions Assistant for Events

SimEvents lets you select from a list of statistical distributions that generate template code for simulating stochastic event actions. Also, you can automatically generate MATLAB code that allows for simulating repeated sequences of event actions.

1   Open a new model and add the Entity server block from the SimEvents library.



2   In the block dialog box, from the **Insert pattern** list, select `Repeating sequence` or `Random number` if you want to insert event action code from a template.
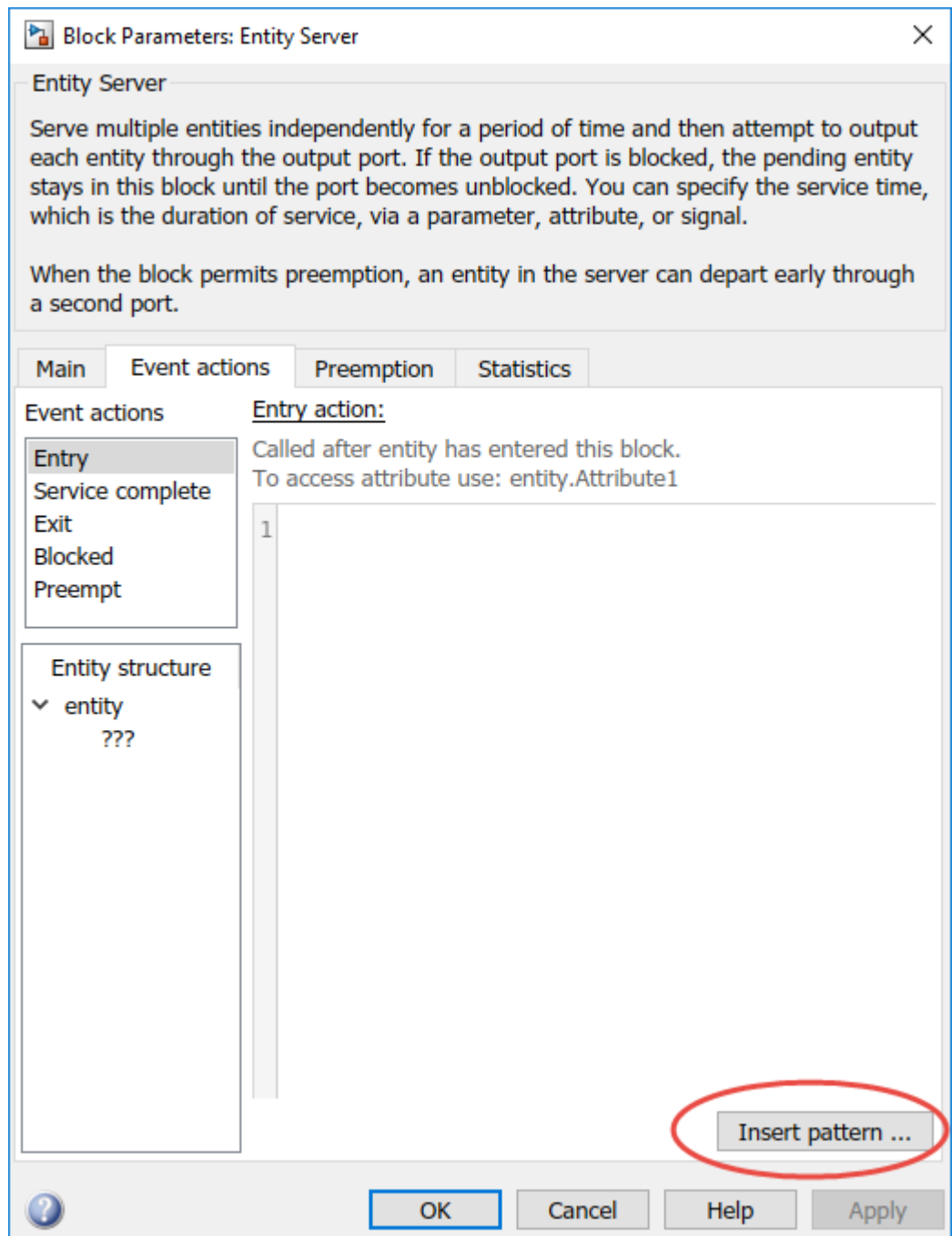
- `Repeating sequence` allows you to:

  - Fix the sequence by setting `Sequence value`
  - Select a **Output after the final value** of the sequence to `Repeat`, `Set to zero`, or `Set to infinity`
  - Select a variable to **Assign output to**
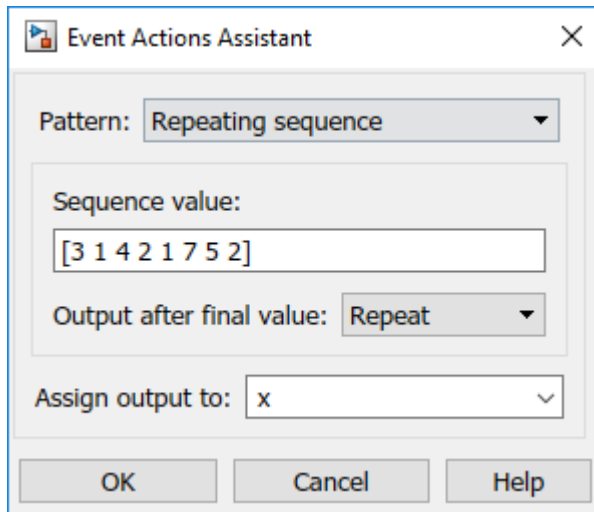


- `Random number` allows you to:

  - Provide an initial value to the random generator engine by setting the `Seed`
  - Select **Distribution** to select from a list of statistical distributions
  - Select a variable to **Assign output to**
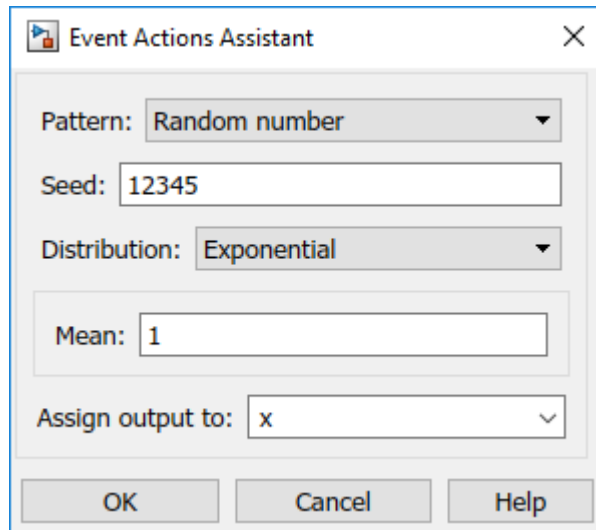
**3** Code is automatically generated in the block dialog box

# See Also

Discrete Event Chart | Entity Generator | Entity Queue | Entity Server | Entity Terminator | MATLAB Discrete Event System | Multicast Receive Queue | Resource Acquirer

## More About

· "Events and Event Actions"

# Run Sample Models

| In this section... |
| --- |
| "Examine Entities and Ports in a Model" on page 1-14 |
| "Entity Appearance" on page 1-16 |
| "Run the Simulation" on page 1-16 |

One way to become familiar with the basics of SimEvents models and the way they work is to examine and run a previously built model. See SimEvents Examples for these examples.

## Examine Entities and Ports in a Model

The `seExampleTankFilling` model illustrates entity and statistics output lines.
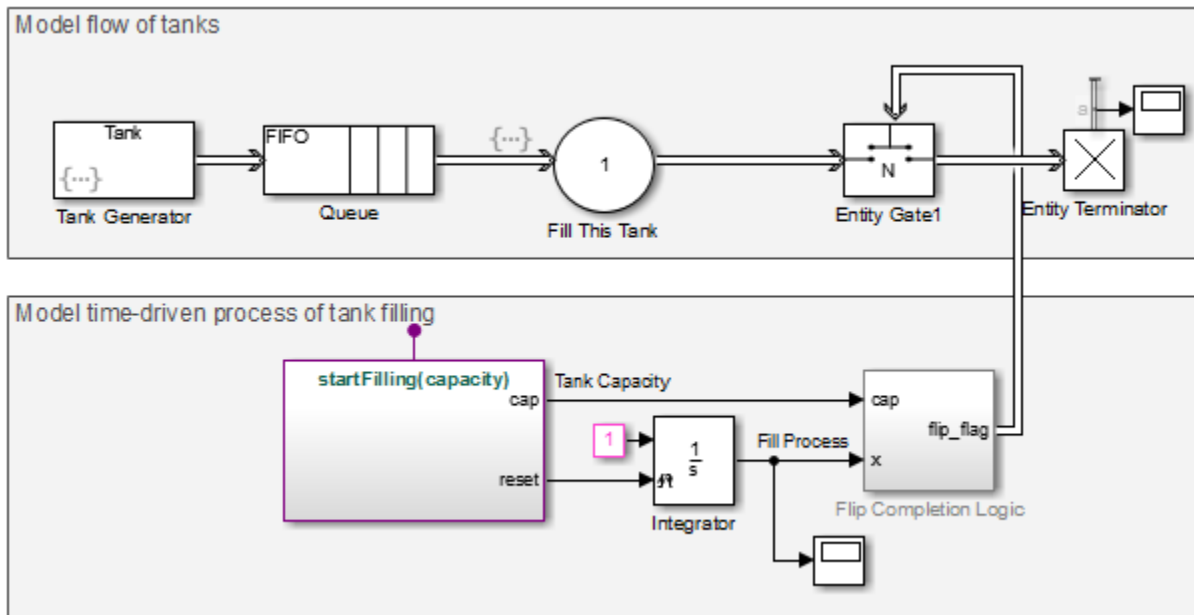


A thick double arrow line indicates the flow of entities.
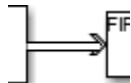
The report of statistics is an important part of the SimEvents blocks. Various SimEvents blocks can report statistics such as:

- Number of entities departed
- Number of entities in the block
- Number of pending entities

When you request a statistic for a block, the output ports for the block extend from the top of the block, with a label for each port. Here is an example of an Entity Terminator block displaying the number of entities arrived (a).



Entity Terminator

To examine a statistic, you can connect the line to a Simulink scope.

**Note** You can also examine the path of entities using the Message Viewer by inserting a Message Viewer block in your model.

### Entity Connections

Some blocks in this model can process entities, discussed in the "What Is an Entity?" on page 1-7 section.

The Entity Generator block and the Entity Queue block, which are part of the SimEvents library set, process entities in this model. Each of these blocks has an entity input port and an entity output port. The figure shows the entity output port of the Entity Generator block and the entity input port of the Entity Queue block.

A thick double arrow line indicates the flow of entities.
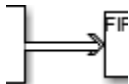


Entity connection lines represent relationships between two blocks (or between their entity ports) by indicating a path by which an entity can:

- Depart from one block
- Arrive simultaneously at a subsequent block

When you run the simulation, entities that depart from the output port arrive simultaneously at the input port.

You cannot branch an entity connection line. If your application requires an entity to arrive at multiple blocks, use the Entity Replicator block to create copies of the entity. In addition, you can route entities using the input and output switch blocks

### Data Connections

A SimEvents model can also use a signal line that represents persistent value-based data such as statistics and states. This line looks like a typical Simulink signal line. Such signals have no associated events and event actions.

## Entity Appearance

Entities do not appear explicitly in the model window. However, you can gather information about entities using Simulink scopes.

## Run the Simulation

To run the `seExampleTankFilling` simulation, choose **Simulation > Run** from the model window.

# See Also

Composite Entity Creator | Composite Entity Splitter | Discrete Event Chart | Entity Gate | Entity Generator | Entity Input Switch | Entity Multicast | Entity Output Switch | Entity Queue | Entity Replicator | Entity Server | Entity Terminator | MATLAB Discrete Event System | Multicast Receive Queue | Resource Acquirer | Resource Pool | Resource Releaser

## Related Examples

- "Simulate a Hybrid System" on page 6-2

## More About

· "SimEvents Common Design Patterns" on page 1-18

# SimEvents Common Design Patterns

The SimEvents library provides design patterns that you can refer to while modeling. To access these patterns, open the SimEvents library and double-click the Design Patterns block.
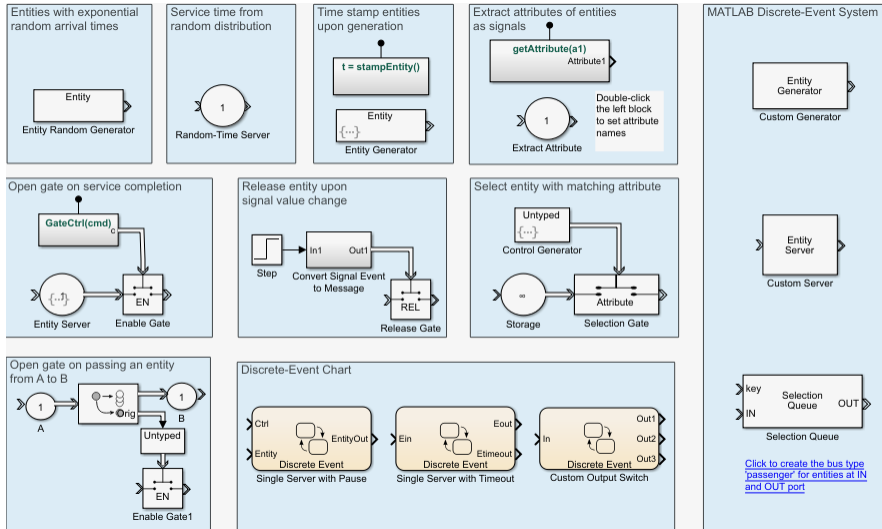


Consider these design patterns while modeling:

| Design Pattern | Description | Input Specifications | Output Specifications | Application |
|---|---|---|---|---|
| Entities with exponential random arrival times | Generates entities with random interval time in exponential distribution fashion. | Not applicable | Structured entity with specified attributes | Model:<br>• Customers entering a store<br>• Incoming phone calls of a hotline |

| Design Pattern | Description | Input Specifications | Output Specifications | Application |
|---|---|---|---|---|
| Service time from random distribution | Specifies waiting time in the Entity Server as a random number uniformly distributed from 0 through 1. | Any entity type | Inherited from the input | Model:<br><br>• Extension of an event that is random within a range (for example, length of a call<br><br>• Purposeful holding of an entity for a random time |
| Extract attributes of entities as signals | Extracts one or more attributes of entities as signals. | A structured entity or bus object with specified attribute | `getAttribute` — Real double scalar signal<br><br>Extracted Attribute — Inherited from the input | Inspect or use a specific entity attribute |
| Timestamp entities upon generation | Generates entities with an attribute **TimeStamp** that records the simulation time upon generation. | Not applicable | Structured entity with attributes `Data` and `TimeStamp` | Use when generation time of entities is needed, for example, when calculating the priority in a combined scheduling algorithm. |

| Design Pattern | Description | Input Specifications | Output Specifications | Application |
|---|---|---|---|---|
| Release entity upon signal value change | Releases an incoming entity when there is a jump in the step function. | Any entity type | Inherited from the input | Use to control the passing of entities based on the change of a function. |
| Open gate on service completion | Upon service completion, the gate opens and releases an entity. | Any entity type | Inherited from the input | Use task completion to trigger entity processing. |
| Sense an entity passing from A to B and open a gate | Passing an entity from *A* to *B* opens the gate and releases an entity. | Any entity type | Inherited from the input | Use to model the passing of an entity in one route to control the passing of another route. |
| Select an entity with a matching attribute | Select entities to advance whose specified attributes are matching the anonymous entity at the control port | A structured entity or bus object with a specified attribute | Inherited from the input | Select entities with a specified attributes to output |
| Discrete Event Chart: Single Server with Pause | A Ctrl message triggers pause of service for the incoming entity. A second Ctrl message continues the service. Entity data conveys the service time. | Ctrl — Anonymous entity specifying the pause and resume<br><br>Entity — Anonymous entity specifying service time | Inherited from the input | Use external events or signals to pause the service of entities. |

| Design Pattern | Description | Input Specifications | Output Specifications | Application |
|---|---|---|---|---|
| Discrete Event Chart: Single Server with Timeout | If the service time (which is random) exceeds the timeout limit specified by the entity data, the entity leaves the server. | Anonymous entity with specified timeout limit | Inherited from the input | Model:<br><br>• A protocol that explicitly calls for timeouts.<br><br>• Implementation of special routing or other handling of entities that exceed a time limit.<br><br>• Entities that represent perishable items. |
| Discrete Event Chart: Custom Output Switch | Randomly routes entities to one of the three output ports. | Anonymous entity | Inherited from the input | Implement a more complicated routing algorithm for an output switch. |

| Design Pattern | Description | Input Specifications | Output Specifications | Application |
|---|---|---|---|---|
| MATLAB Discrete Event System: Custom Generator | The Custom Generator block, defined using the MATLAB Discrete Event System block, is a basic entity generator. The generator block requires specification of generation period. | Not applicable | Anonymous entity | Implement a more complicated entity generator. |
| MATLAB Discrete Event System: Custom Server | Custom Server block, defined using the MATLAB Discrete Event System block, is a basic entity server. The server block requires specification of server number and service time. | Any entity type | Inherited from the input | Implement a more complicated entity server. |

| Design Pattern | Description | Input Specifications | Output Specifications | Application |
|---|---|---|---|---|
| MATLAB Discrete Event System: Selection Queue | The Selection Queue block, defined using the MATLAB Discrete Event System block, stores entities of bus type `passenger` arriving at the IN port. Keys from the call port select `passenger` entities with the matching `trainNum` field and send them to the OUT port. | Key — Anonymous entity carrying the selection key<br><br>IN — A structured entity or bus object with specified attribute | Inherit from IN | Select a specific entity to output from a queue. |

# See Also

Composite Entity Creator | Composite Entity Splitter | Discrete Event Chart | Entity Gate | Entity Generator | Entity Input Switch | Entity Multicast | Entity Output Switch | Entity Queue | Entity Replicator | Entity Server | Entity Terminator | MATLAB Discrete Event System | Multicast Receive Queue | Resource Acquirer | Resource Pool | Resource Releaser

## Related Examples

- "Run Sample Models" on page 1-14

## More About

- "Discrete-Event Simulation in Simulink Models" on page 1-3

# Build Simple Models with SimEvents Software

# Build a Discrete-Event Model

| In this section... |
|---|
| "Open a Model and Library" on page 2-2 |
| "Move Blocks into the Model Window" on page 2-3 |
| "Configure Blocks" on page 2-4 |
| "Connect Blocks" on page 2-7 |
| "Run the Simulation" on page 2-7 |

This example describes how to build a new model representing a discrete-event system. The system is a simple queuing system in which "customers" — entities — arrive at a fixed deterministic rate, wait in a queue, and advance to a server that operates at a fixed deterministic rate. This type of system is known as a D/D/1 queuing system in queuing notation. The notation indicates a deterministic arrival rate, a deterministic service rate, and a single server.

The example system shows how to perform basic model-building tasks, such as:

- Adding blocks to models
- Configuring blocks using their parameter dialog boxes

## Open a Model and Library

The first steps in building a model are to set up your environment, open a new model on page 2-2 window, and open the libraries on page 2-2 containing blocks.

### Open a New Model Window

On the **Home** tab, select **New** > **Simulink Model**. An empty model window opens.

To name the model and save it as a file, select **File** > **Save** from the model window's menu. Save the model in your working folder as dd1.

### Open the SimEvents Library

In the MATLAB Command Window, enter

```
simevents
```

The main SimEvents library window appears. This window contains an icon for each SimEvents library. To open a library and view the blocks it contains, double-click the icon that represents that library.
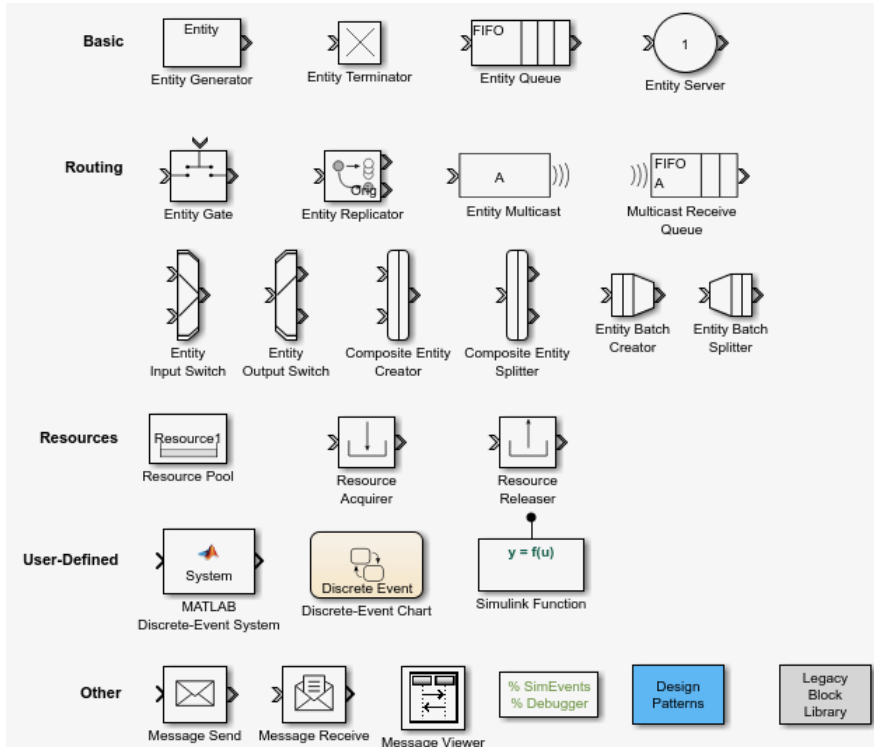


### Open Simulink Libraries

In the Simulink editor, click the Library Browser button. The Simulink Library Browser opens, using a tree structure to display the available libraries and blocks. To view the blocks in a library listed in the left pane, select the library name, and the list of blocks appears in the right pane. The Library Browser provides access not only to Simulink blocks but also to SimEvents blocks.

## Move Blocks into the Model Window

To move blocks from libraries into the model window, follow these steps:

1 In the SimEvents library window, drag the Entity Generator block from the library into the model window.

2 Drag the Entity Queue block into the model window.

3 Drag the Entity Server block into the model window.

4 Drag the Entity Terminator block into the model window.

5 From the Simulink Sinks library, drag the Scope block into the model window.

As a result, the model window contains blocks that represent the key processes in the simulation: blocks that generate entities, store entities in a queue, serve entities, and create a plot showing relevant data.
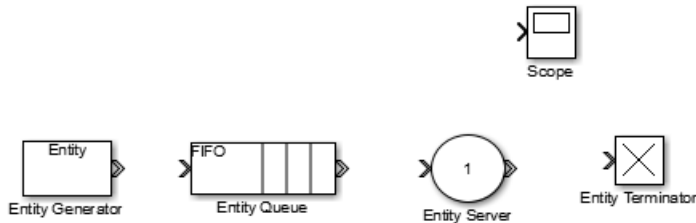


## Configure Blocks

Each block in a model, in this case, `dd1`, has a dialog box that enables you to specify block parameters. Default parameter values might or might not be appropriate, depending on what you are modeling.

### View Parameter Values

Two important parameters in the D/D/1 queuing system are the arrival rate and service rate. The reciprocals of these rates are the duration between successive entities and the duration of service for each entity. To examine these durations:

1 Double-click the Entity Generator block to open its dialog box. Observe and that the **Period** parameter is set to `1`. This means that the block generates a new entity every second.

2 Double-click the Entity Server block to open its dialog box. Observe that the **Service time** parameter is set to `1.0`. This means that the server spends one second processing each entity that arrives at the block.

3 Click **Cancel** in both dialog boxes to dismiss them without changing any parameters.

The **Period** and **Service time** parameters have the same value, which means that the server completes an entity's service at exactly the same time that a new entity is being created.

### Change Parameter Values

Configure blocks to create a plot that shows when each entity departs from the server, and to make the queue have an infinite capacity.

1  Double-click the Entity Server block to open its dialog box.

2  Click the **Statistics** tab to view parameters related to the statistical reporting of the block.

3  Select **Number of entities departed, d**.
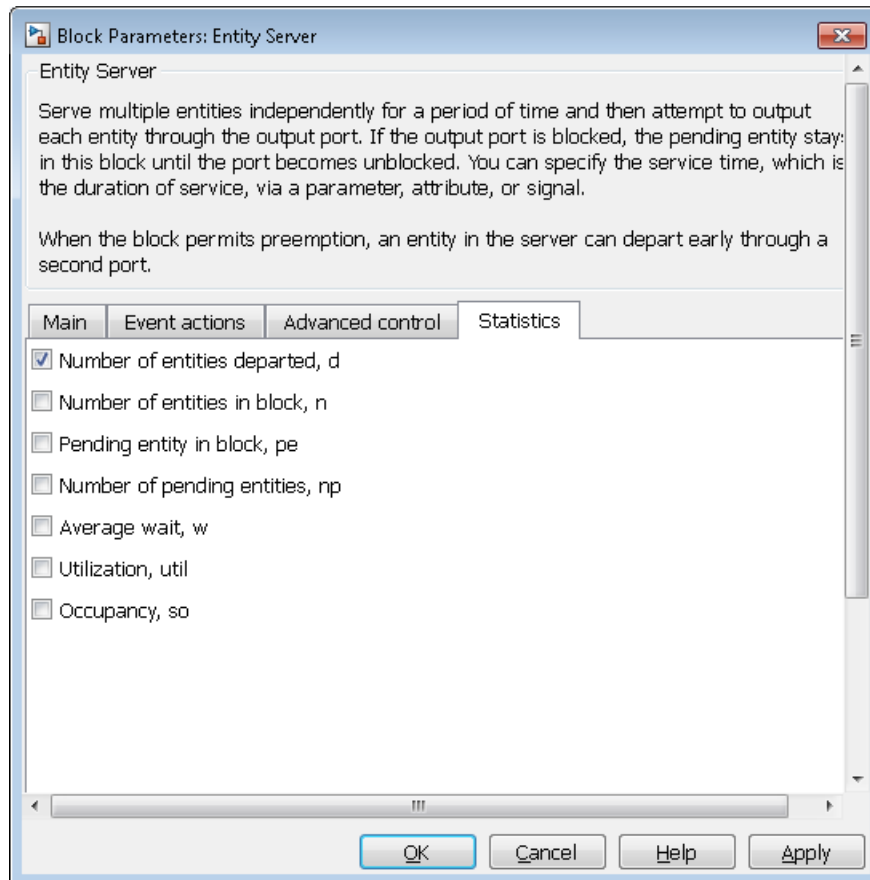
Then click **OK**. The Entity Server block acquires a signal output port labeled **d**. During the simulation, the block will produce an output signal at this **d** port; the signal's value is the running count of entities that have completed their service and departed from the server. You can connect a Scope block to this entity line and display the statistics (running count of entities).

4   Double-click the Entity Queue block to open its dialog box.

5   Set the **Capacity** parameter to `Inf` and click **OK**.

The model window now contains blocks that represents key processes.

## Connect Blocks

Connect the blocks as shown and save the dd1 model you have created.
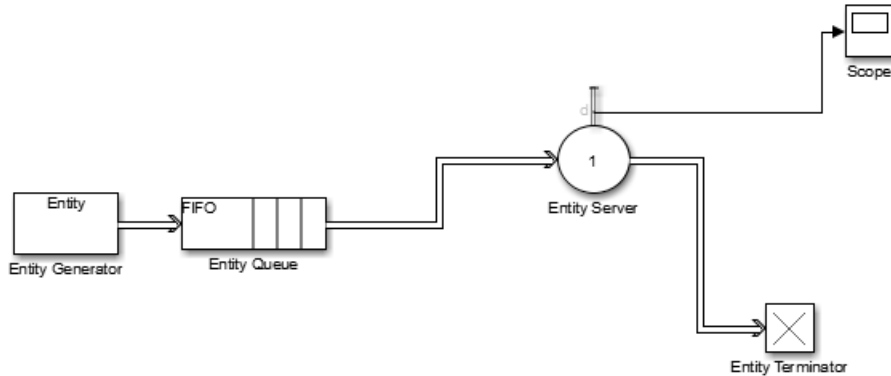


SimEvents connects the source block to the destination block. If necessary, the software also routes the connecting line around intervening blocks or lines.

## Run the Simulation

To start the simulation, select **Simulation** > **Run** from the model window's menu.

### Results of the Simulation

When the simulation runs, the Simulink Scope block opens a window containing a plot. The horizontal axis represents the times at which entities depart from the server, while the vertical axis represents the total number of entities that have departed from the server.
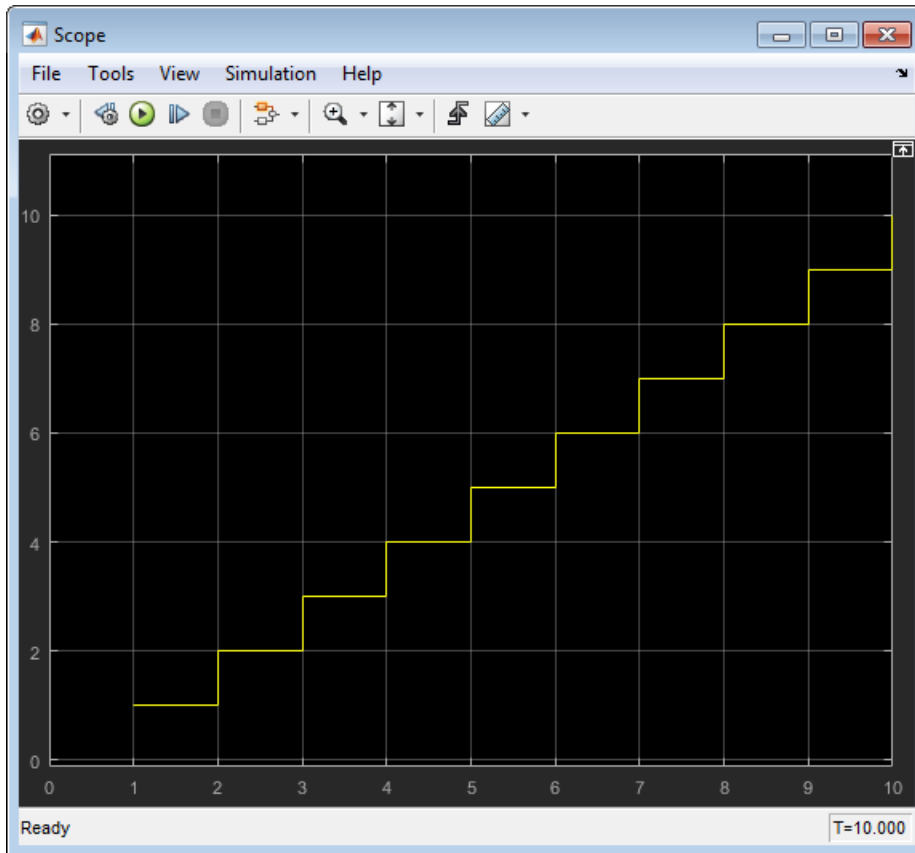
After an entity departs from the Entity Server block, the block updates its output signal at the **d** port.

## See Also

Composite Entity Creator | Composite Entity Splitter | Discrete Event Chart | Entity Gate | Entity Generator | Entity Input Switch | Entity Multicast | Entity Output Switch | Entity Queue | Entity Replicator | Entity Server | Entity Terminator | MATLAB Discrete Event System | Multicast Receive Queue | Resource Acquirer | Resource Pool | Resource Releaser

## Related Examples

- "Explore Simulation Results Using Plots" on page 2-10
- "Simulate a Hybrid System" on page 6-2

## More About

- "What Is an Entity?" on page 1-7
- "Role of Entities in SimEvents Models" on page 3-2
- "What Is an Event?" on page 1-9
- "Storage" on page 3-13
- "Write Events Actions" on page 3-22
- "Statistics Through SimEvents Blocks" on page 4-2

# Explore Simulation Results Using Plots

| In this section... |
|---|
| "Explore the D/D/1 System Using Plots" on page 2-10 |
| "Visualize and Animate Simulations" on page 2-12 |
| "Explore the System Using the Simulink Simulation Stepper" on page 2-13 |
| "Information About Race Conditions and Random Times" on page 2-13 |

## Explore the D/D/1 System Using Plots

The `dd1` model that you created in "Build a Discrete-Event Model" on page 2-2 plots the number of entities that depart from the server. This section modifies the model to plot other quantities that can reveal aspects of the simulation. The topics are as follows:

- "View Statistics for Waiting Times and Utilization" on page 2-10
- "Observations from Plots" on page 2-11

### View Statistics for Waiting Times and Utilization

The queue length is an example of a statistic that quantifies a state at a particular instant. Other statistics, such as average waiting time and server utilization, summarize behavior between simtime=0 and the current time. To modify the model so that you can view the average waiting time of entities in the queue and server, as well as the proportion of time that the server spends storing an entity, use the following procedure:

1. Double-click the Entity Queue block to open its dialog box. Click the **Statistics** tab, set the **Average wait** parameter to `On`, and click **OK**. This causes the block to have a signal output port for the signal representing the average duration that entities wait in the queue. The port label is **w**.

2. Double-click the Entity Server block to open its dialog box. Click the **Statistics** tab, set both the **Average wait** and **Utilization** parameters to `On`, and click **OK**. This causes the block to have a signal output port labeled **w** for the signal representing the average duration that entities wait in the server, and a signal output port labeled **u** for the signal representing the proportion of time that the server spends storing an entity.

3. Copy the Scope1 block and paste it into the model window.

**4**    Double-click the new copy to open its dialog box.

**5**    Copy the Scope2 block that you just modified and paste it into the model window twice. You now have four scope blocks.

      Each copy assumes a unique name. If you want to make the model and plots easier to read, you can click the names underneath each scope block and rename the block to use a descriptive name like Queue Waiting Time, for example.

**6**    Connect the **u** signal output port and the two **w** signal output ports to the **in** signal input ports of the unconnected scope blocks by dragging the mouse pointer from port to port.
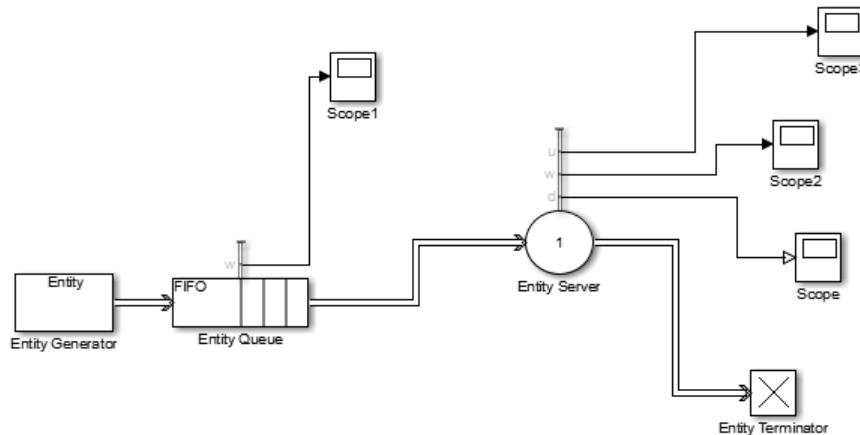


**7**    Save the model.

**8**    Run the simulation with different values of the **Period** parameter in the Entity Generator block. Look at the plots to see how they change if you set the intergeneration time to `0.3` or `1.1`, for example.

---

**Note**   Scope blocks do not support bus objects. SimEvents software supports Scope blocks with only single inputs.

---

### Observations from Plots

- The average waiting time in the server does not change after the first departure from the server because the service time is fixed for all departed entities. The average

waiting time statistic does not include partial waiting times for entities that are in the server but have not yet departed.

- The utilization of the server is nondecreasing if the intergeneration time is small (such as `0.3`) because the server is constantly busy once it receives the first entity.

  The utilization might decrease if the intergeneration time is larger than the service time (such as `1.5`) because the server has idle periods between entities.

- The average waiting time in the queue increases throughout the simulation if the intergeneration time is small (such as `0.3`) because the queue gets longer and longer.

  The average waiting time in the queue is zero if the intergeneration time is larger than the service time (such as `1.1`) because every entity that arrives at the queue is able to depart immediately.

## Visualize and Animate Simulations

You can explore the following elements of a SimEvents model with these tools.

| Items to Observe | Visualization Tool |
|---|---|
| Statistics | • Simulation Data Inspector |
| Entities passing through model | • Simulink To Workspace block |
| | • Simulink Scope block |
| | • Simulink Display block |
| | • Simulink To File block |
| | • Simulink dashboard blocks |
| Entity animation | **Display** > **Message Animation** |
| Step through Simulation | Simulink Simulation Stepper |
| Custom animation | Use SimEvents custom visualization API. |

The Simulink Floating Scope does not support SimEvents models.

Simulation Data Inspector is a powerful and unified user interface for viewing both entities and signal (for example, statistics) data. For more information, see "Inspect and Analyze Simulation Results" (Simulink).

### Animate Simulations

During simulation, animation provides visual verification that your model behaves as you expect. Animation highlights active entities in a model as execution progresses. You can control the speed of entity activity animation during simulation, or turn animation off. In the Simulink editor, select **Display** > **SimEvents Animation Menu**, then select:

- **Fast**
- **Medium**
- **Slow**
- **None**

The **Fast** animation speed shows the active highlights at each time step. To add delay with each time step, set the animation speed to **Medium** or **Slow**. To turn off animation, in the Simulink editor, select **Display** > **Message Animation** > **None**.

Animation is disabled by default in SimEvents models.

## Explore the System Using the Simulink Simulation Stepper

Simulation Stepper enables you to step through major time steps of a simulation. Use this tool to explore your discrete-event system. For more information, see "Simulation Stepper" (Simulink).

## Information About Race Conditions and Random Times

You can vary the processing sequence for simultaneous events or make the intergeneration times or service times random.

# See Also

Entity Queue

## Related Examples

- "Build a Discrete-Event Model" on page 2-2

# Key Concepts in SimEvents Software

# Role of Entities in SimEvents Models

Entities are discrete items of interest in a discrete-event simulation. You determine what an entity signifies, based on what you are modeling. For more information, see "What Is an Entity?" on page 1-7.

SimEvents models typically contain at least one source block that generates entities. Other SimEvents blocks in the model process the entities that the source block generates.

## Meaning of Entities in Different Applications

An entity represents an item of interest in a discrete-event simulation. The meaning of an entity depends on what you are modeling. In this topic, examples use entities to represent abstract customers in a queuing system and data packets from a remote controller to an actuator on the system being controlled.

Entities do not have a graphical depiction in the model window the way blocks, ports, and connection lines do.

## Vary the Interpretation of Entities

A single model can use entities to represent different kinds of items. For example, if you are modeling a factory that processes two different kinds of parts, you can:

- Use two Entity Generator blocks to create the two kinds of parts.
- Use one Entity Generator block and subsequently assign an attribute to indicate what kind of part each entity represents.

SimEvents entities are fundamentally the same as Stateflow messages.

## Data and Entities

You can optionally attach data to entities. Such data is stored in one or more attributes of an entity. You define names and numeric values for attributes. For example, if your entities represent a message that you are transmitting across a communication network, you might assign data called `length` that indicates the length of each particular message. You can read or change the values of attributes during the simulation.

## Data and Signals

In SimEvents models, signals carry persistent value-based information such as statistics and states.

## Introduction to Time-Based Entities

By default, the Entity Generator block creates time-based entities. Change the **Time Source** parameter to select the time source for the entity generation. You can create time-based entities using:

- The **Period** parameter value. For more information, see "Create a Time-based Entity" on page 3-4.
- A signal port. You can then connect a Simulink source block, such as a Repeating Sequence block, to the signal port. The time value cannot be a negative number.
- MATLAB code. For more information, see "Create Randomized Entities" on page 3-5.

## Role of Attributes in Models

You can optionally attach data to entities. Such data is stored in one or more attributes of an entity. You define names and numeric values for attributes. For example, if entities represent a message you are transmitting across a communication network, you might also assign data called `length` that indicates the length of each particular message. You can read or change the values of attributes during the simulation.

You can optionally specify the structure of an entity using a Simulink bus object. This capability is useful when defining complex entity structures that need to be defined once,

but used in multiple locations in a model. In addition, the MATLAB Discrete-Event System and Discrete Event Chart blocks require that you specify entities as bus objects. For more information on bus objects, see "When to Use Bus Objects" (Simulink).

## Create Entities

There are many ways to create entities. Two common ways are using a time-based method, and using a random number generator.

### Create a Time-based Entity

Use the Entity Generation block to create time-based entities. The Entity Generation lets you specify a period at which it creates entities.

1  Open the SimEvents block library. You can use the Simulink browser or type `simevents` at the MATLAB Command Window.

2  Create a new model.

3  From the SimEvents library, drag the Entity Generator block to the new model.

4  From the SimEvents library, drag the Entity Queue block to the new model.

   - Connect the Entity Generator block to the input of the Entity Queue.

   - In the Entity Queue block, select **Number of entities departed, d**.

5  From the Simulink Sinks library, drag a Scope block to the new model. Connect the Scope block to the d port of the Entity Queue block.

6  From the SimEvents library, drag an Entity Terminator block to the new model. Connect the output of the Entity Queue block to the input of the Entity Terminator block.
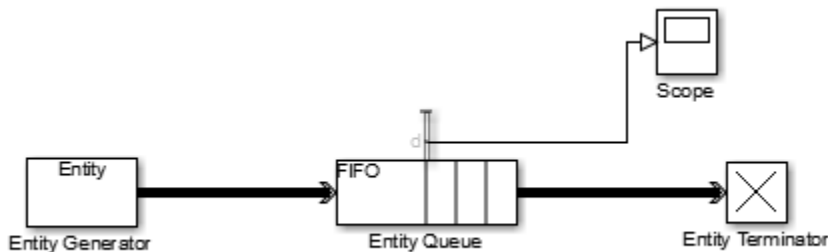


Upon simulation, the scope displays the entities that depart the queue.
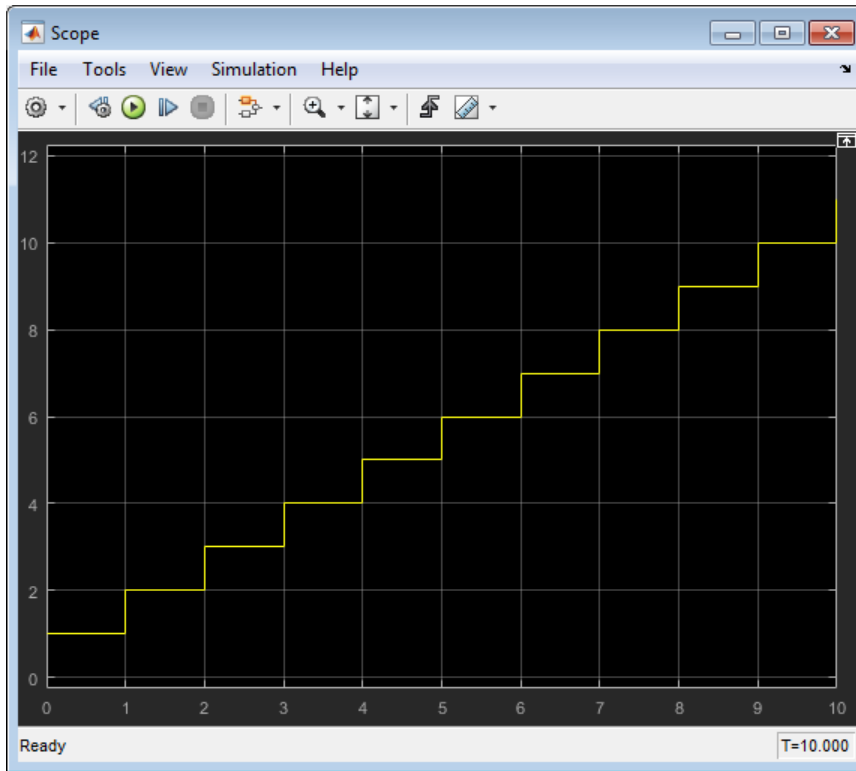
**Note** You cannot connect a scope to a SimEvents line, as denoted by the thick double arrow line.

### Create Randomized Entities

Use the Entity Generation block to create time-based entities. The Entity Generation lets you specify a randomization operation (such as the MATLAB `rand` function) to create entities at random times.

1 Open the SimEvents block library. You can use the Simulink browser or type `simevents` at the MATLAB Command Window.

2 Create a new model.

3 From the SimEvents library, drag the Entity Generator block to the new model.

**a** Double-click the block and set the **Time source** parameter to `MATLAB action`.

**b** In the **Intergeneration time action** parameter, enter a call to a randomizer function, such as `rand`. For example:

```
dt = rand(1,1);
```
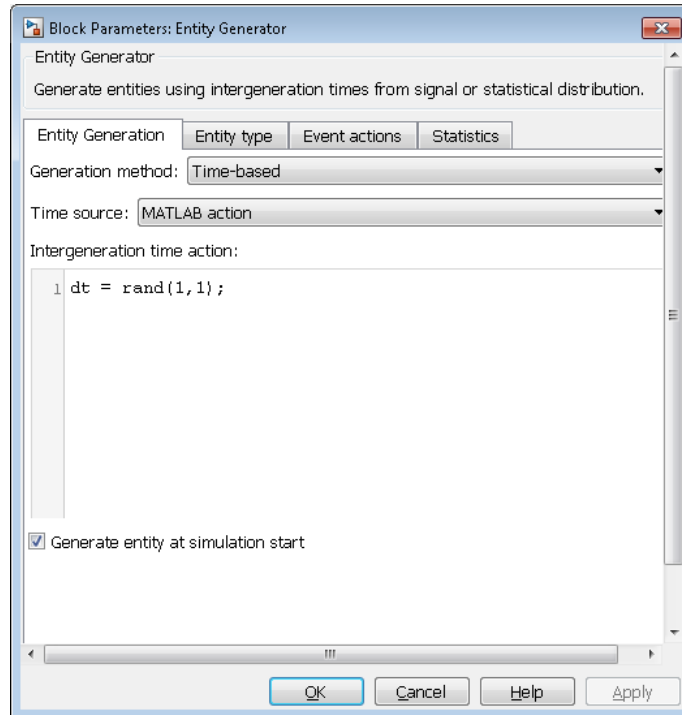


**4** From the SimEvents library, drag the Entity Queue block to the new model.

- Connect the Entity Generator block to the input of the Entity Queue
- In the Entity Queue block, select **Number of entities departed, d**.

**5** From the Simulink Sinks library, drag a Scope block to the new model. Connect the Scope block to the d port of the Entity Queue block.

**6** From the SimEvents library, drag an Entity Terminator block to the new model. Connect the output of the Entity Queue block to the input of the Entity Terminator block.

Upon simulation, the scope displays the entities that depart the queue.

# See Also

Composite Entity Creator | Composite Entity Splitter | Discrete Event Chart | Entity Gate | Entity Generator | Entity Input Switch | Entity Multicast | Entity Output Switch | Entity Queue | Entity Replicator | Entity Server | Entity Terminator | MATLAB Discrete Event System | Multicast Receive Queue | Resource Acquirer | Resource Pool | Resource Releaser

## Related Examples

- "Generate Entities When Events Occur"
- "Specify Intergeneration Times for Entities"
- "Manipulate Entity Attributes"
- "Inspect Structures of Entities"

## More About

- "Entity Types"
- "Attribute Value Support"

# Role of Entity Ports and Paths

## Entity Ports and Paths

An entity output port provides a way for an entity to depart from a block. Conversely, an entity input port provides a way for an entity to arrive at a block.

A connection line indicates a path along which an entity can potentially advance. However, the connection line does not imply that any entities actually advance along that path during a simulation. For a given entity path and a given time instant during the simulation, any of the following could be true:

- No entity is trying to advance along that path.
- An entity has tried and failed to advance along that path. For some blocks, it is normal for an entity input port to be unavailable under certain conditions. As a result, the entity fails in its attempt to advance along that path, even though the path is intact (that is, even though the ports are connected). An entity that tries and fails to advance is called a pending entity.
- One or more entities successfully advance along that path. This occurs only at a discrete set of times during a simulation.

**Note** The simulation could also have one or more times at which one or more entities successfully advance along a given entity path. Simultaneously, one or more different entities try and fail to advance along that same entity path. For example, an entity departs from a queue and, simultaneously, the next entity in the queue tries and fails to depart.

## Definition of Entity Paths

An entity path is a connection from an entity output port to an entity input port, depicted as a line connecting the entity ports of two SimEvents blocks. An entity path represents

the equivalence between an entity's departure from the first block and arrival at the second block. For example, in the model shown below, any entity that departs from the Entity Queue block's output port equivalently arrives at the Entity Server block's input port.
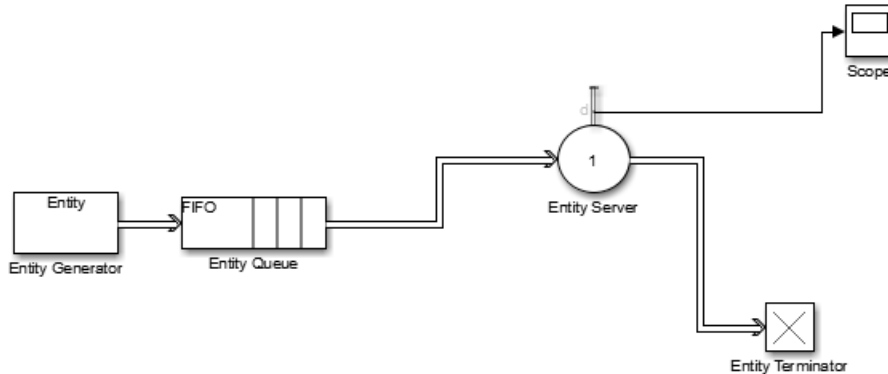


The existence of the entity path does not guarantee that any entity actually uses the path. For example, the simulation could be so short that no entities are ever generated. Even when an entity path is used, it is used only at a discrete set of times during the simulation.

## Implications of Entity Paths

In some models, you can use the entity connection lines to infer the full sequence of blocks at which a given entity arrives, throughout the simulation.

In many discrete-event models, however, the set of entity connection lines does not completely determine the sequence of blocks at which each entity arrives. This example shows two queues in a parallel arrangement, preceded by a block that has one entity input port and two entity output ports.
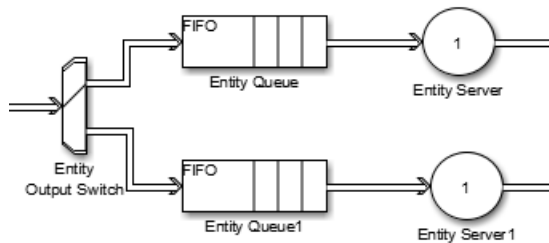
By looking at the entity connection lines alone, you cannot tell which queue block's **IN** port an entity will arrive at. Instead, you need to know more about how the one-to-two block (Output Switch) behaves and understand the outcome of certain run-time decisions.

## Designing Paths Using Input, Output, and Entity Combiner Blocks

You design entity paths by choosing or combining entity paths using the Entity Input Switch, Entity Output Switch, and Entity Combiner blocks of the SimEvents library. These blocks have extra entity ports that let you vary the model's topology (that is, the set of blocks and connection lines).

Typical reasons for manipulating entity paths are:

- To describe an inherently parallel behavior in the situation you are modeling — for example, a computer cluster with two computers that share the computing load. You can use the Entity Output Switch block to send computing jobs to one of the two computers. You might also use the Entity Input Switch block if computing jobs share a common destination following the pair of computers.
- To design nonlinear topologies, such as feedback loops — for example, repeating an operation if quality criteria such as quality of service (QoS) are not met. You can use the Entity Input Switch block with the **Active port selection** parameter set to `All` to combine the paths of new entities and entities that require a repeated operation.
- To incorporate logical decision-making into your simulation — for example, to determine scheduling protocols. You might use the Entify Input Switch block to determine which of several queues receives attention from a server.
- To incorporate logic for activation or deactivation of an entity path, use the Entity Gate block. For example, you can activate an entity path for one entity when a condition is fulfilled in your model.
- To model routing of copies of an entity to multiple remote locations in the model, consider using the Entity Multicast and Multicast Receive Queue blocks.

Other libraries in the SimEvents library set contain a number of blocks whose secondary features, such as preemption from a server or timeout from a queue or server, give you opportunities to design paths.

# See Also
Entity Input Switch | Entity Output Switch

## Related Examples

- "Select Departure Path Using Entity Output Switch"
- "Select Arrival Path Using Entity Input Switch"
- "Combine Entity Paths"
- "Use Messages To Route Entities"

## More About

- "Role of Paths in SimEvents Models"
- "Use Attributes to Route Entities"
- "Role of Gates in SimEvents Models"

# Storage

| In this section... |
| --- |
| "Queues and Servers" on page 3-13 |
| "Behavior and Features of Queues" on page 3-13 |
| "Physical Queues and Logical Queues" on page 3-14 |
| "Queue Policies" on page 3-14 |
| "Storage Actions" on page 3-14 |
| "Behavior and Features of Servers" on page 3-16 |
| "What Servers Represent" on page 3-16 |
| "Common Server Use Cases" on page 3-17 |
| "Constructs Involving Queues and Servers" on page 3-17 |
| "Broadcast Entities Using Multicast Mode" on page 3-18 |
| "Entity Resources" on page 3-20 |

## Queues and Servers

Queue and server blocks are storage blocks that hold entities.

- Queues order entities and sort them according to queue policies.
- Servers delay entities until certain conditions are met.

## Behavior and Features of Queues

In a discrete-event simulation, a queue stores entities for a length of time that cannot be determined in advance. The queue attempts to output entities as soon as it can, but its success depends on whether the next block accepts new entities. An everyday example of a queue is when you stand in a line with other people to wait for some type of service to address your needs and you cannot determine in advance how long you must wait.

Distinguishing features of different queues include:

- Capacity — The number of entities the queue can store simultaneously
- Discipline — A feature determines which entity departs first if the queue stores multiple entities

## Physical Queues and Logical Queues

In some cases, a queue in a model is similar to an analogous aspect of the real-world system being modeled. This kind of queue is sometimes called a physical queue. For example, you might use a queue to represent a sequence of:

- People standing in line
- Airplanes waiting to access a runway
- Messages waiting to be sent
- Parts waiting to be assembled in a factory
- Computer programs waiting to be executed

In other cases, a queue in a model does not arise in an obvious way from the real-world system but instead is included for modeling purposes. This kind of queue is sometimes called a logical queue. For example, you might use a queue to provide a temporary storage area for entities that might otherwise have nowhere to go. Such use of a logical queue can prevent deadlocks or simplify the simulation.

Use the Entity Queue block to model queues.

## Queue Policies

The Entity Queue block uses these queue policies:

- FIFO — The block processes the entity as first in first out.
- LIFO — The block processes the entity as last in first out.

- Priority — The block reads the priority from the **Priority Source** parameter. This parameter is a particular attribute value that the block stores based on the value of the number.

## Storage Actions

Storage blocks have event actions based on events influencing entities in the corresponding storage blocks. Each block has a set of actions particular to the block.

| Entity Generator | Entity Queue | Entity Server | Entity Terminator | Resource Acquirer |
|---|---|---|---|---|
| Entity generation | Entity entry to queue block | Entity entry to server block | Entity entry to terminator block | Entity entry to acquirer block |
| Entity exit from block | Entity exit from block | Service completion of entity | N/A | Entity exit from acquirer block |
| N/A | Entity is blocked | Entity exit from block | N/A | N/A |
| N/A | N/A | Entity is blocked | N/A | N/A |
| N/A | N/A | Entity is preempted | N/A | N/A |

This illustration shows the flow of actions as entities move through a discrete-event system simulation.
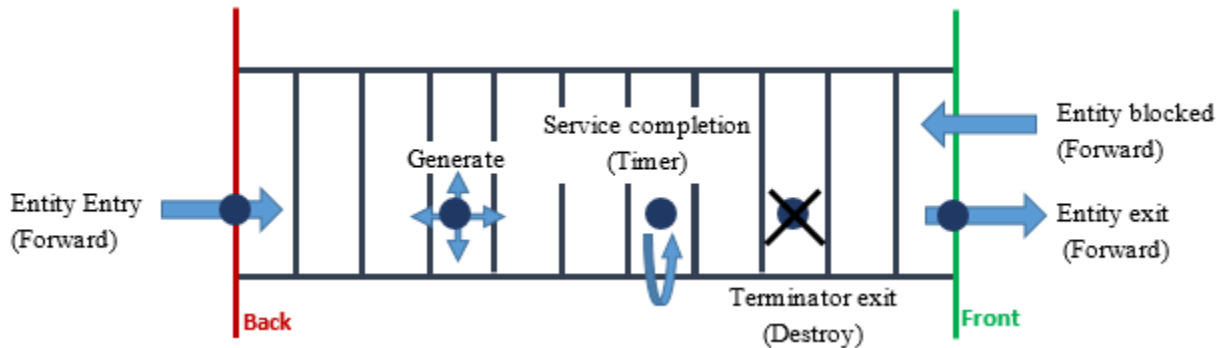


Notes:

- Entity entry, exit, and blocking actions are performed as part of an entity forward event.
- Service completion action is performed following a timer event.
- Entity termination event performs a destruction action.

For more information on event actions, see "Events and Event Actions".

## Behavior and Features of Servers

In a discrete-event simulation, a server stores entities for a length of time, called the service time, and then attempts to output the entity. During the service period, the block is said to be serving the entity that it stores. An everyday example of a server is a person (a bank teller, a retail cashier, etc.) with whom you perform a transaction with a projected duration.

The service time for each entity is computed when it arrives, which contrasts with the inherent unknowability of the storage time for entities in queues. If, however, the next block does not accept the arrival of an entity that has completed its service, the server is forced to hold the entity longer.

Distinguishing features of different servers include:

- The number of entities it can serve simultaneously, which could be finite or infinite
- Characteristics of, or the method of computing, the service times of arriving entities
- Whether the server permits certain arriving entities to preempt entities that are already stored in the server

**Tip** In the absence of preemption, a finite-capacity server does not accept new arrivals when it is already full. You can place a queue before each finite-capacity server, establishing a place for entities to stay while they are waiting for the server to accept them. Otherwise, the waiting entities might be stored in various different locations in the model and the situation might be more difficult for you to predict or analyze.

## What Servers Represent

In some cases, a server in a model is similar to an analogous aspect of the real-world system being modeled. For example, you might use a server to represent:

- A person (such as a bank teller) who performs a transaction with each arriving customer
- A transmitter that processes and sends messages
- A machine that assembles parts in a factory
- A computer that executes programs

### Servers Inserted for Modeling Purposes

In some cases, a server in a model does not represent a real-world system. A common modeling technique involves a delay of duration zero, that is, an infinite server whose service time is zero, to provide a place for an entity to reside to manipulate its attributes.

Use the Entity Server block to model queues.

## Common Server Use Cases

Common server use cases of a server include:

- In a production line application, the processing unit
- In a network application, the processor

## Constructs Involving Queues and Servers

You can combine Entity Queue and Entity Server blocks to model different situations:

- "Serial Queue-Server Pairs" on page 3-17
- "Parallel Queue-Server Pairs as Alternatives" on page 3-18
- "Parallel Queue-Server Pairs in Multicasting" on page 3-18
- "Serial Connection of Queues" on page 3-18
- "Parallel Connection of Queues" on page 3-18

### Serial Queue-Server Pairs

Two queue-server pairs connected in series represent successive operations that an entity undergoes. For example, parts on an assembly line are processed sequentially by two machines.

You can alternatively model the situation as a pair of servers without a queue between them. However, the absence of the queue means that if the first server completes service on an entity before the second server is available:

- The entity must stay in the first server past the end of service.
- The first server cannot accept a new entity for service until the second server becomes available.

### Parallel Queue-Server Pairs as Alternatives

Two queue-server pairs connected in parallel, in which each entity arrives at one or the other, represent alternative operations. For example, vehicles wait in line for one of several tollbooths at a toll plaza. In this case, the model must have decision logic, possibly in the form of a switch preceding this pattern.

### Parallel Queue-Server Pairs in Multicasting

Two queue-server pairs connected in parallel, in which a copy of each entity arrives at both, represent a multicasting situation such as sending a message to multiple recipients. Note that copying entities might not make sense in some applications.

### Serial Connection of Queues

Two queues connected in series might be useful if you are using entities to model items that physically experience two distinct sets of conditions while in storage. For example, additional inventory items that overflow one storage area have to stay in another storage area in which a less well-regulated temperature affects the items' long-term quality. Modeling the two storage areas as distinct queue blocks facilitates viewing the average length of time that entities stay in the overflow storage area.

### Parallel Connection of Queues

Two queues connected in parallel, in which each entity arrives at one or the other, represent alternative paths for waiting. The paths might lead to different operations, such as a line of vehicles waiting for a tollbooth and a line of vehicles waiting on a jammed exit ramp of the freeway. You might model the tollbooth as a server and the traffic jam as a gate.

## Broadcast Entities Using Multicast Mode

Multicast mode enables multiple queues to receive entities from one Entity Multicast block. The receiving block for an Entity Multicast blocks is a Multicast Receive Queue block whose **Tag** parameters have the same value. The Multicast Receive Queue block is essentially the Entity Queue block with the **Entity Arrival source** parameter set to `Multicast`.

Using the Entity Multicast block requires no connecting lines. The **Tag** parameters just need to match.

**1** From the SimEvents library, drag the Entity Multicast and Multicast Receive Queue blocks.

**2** In both dialog boxes, in the **Multicast tag** parameters, enter the same text. For example, A.
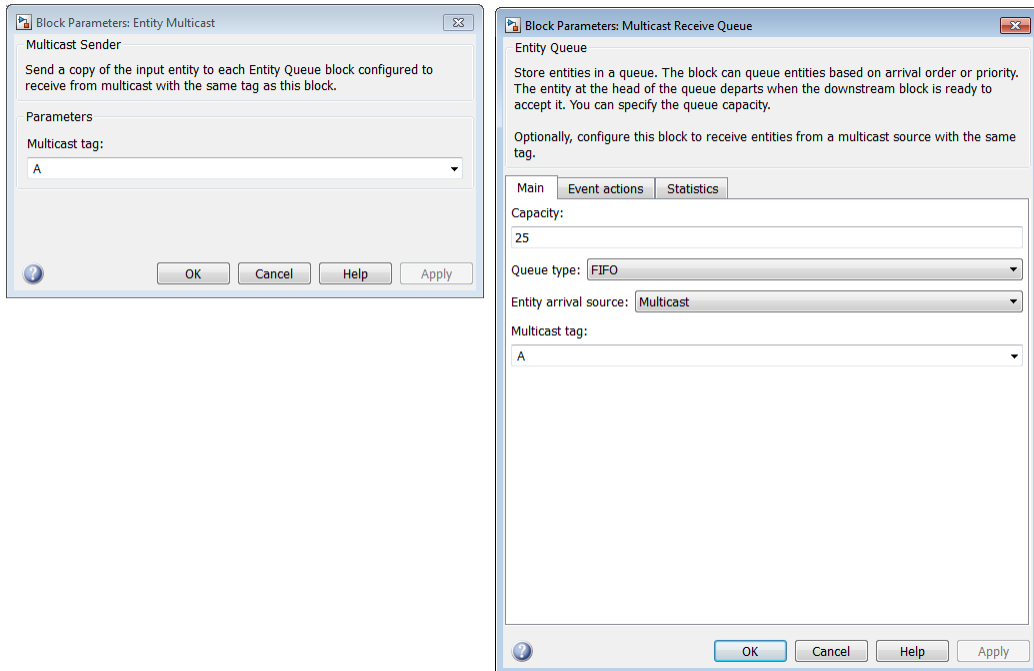


The software uses these tags to match the broadcaster and broadcastees.

This is example shows entities broadcast to two queues. Notice that the FIFO blocks for both queues have the **A** tag.
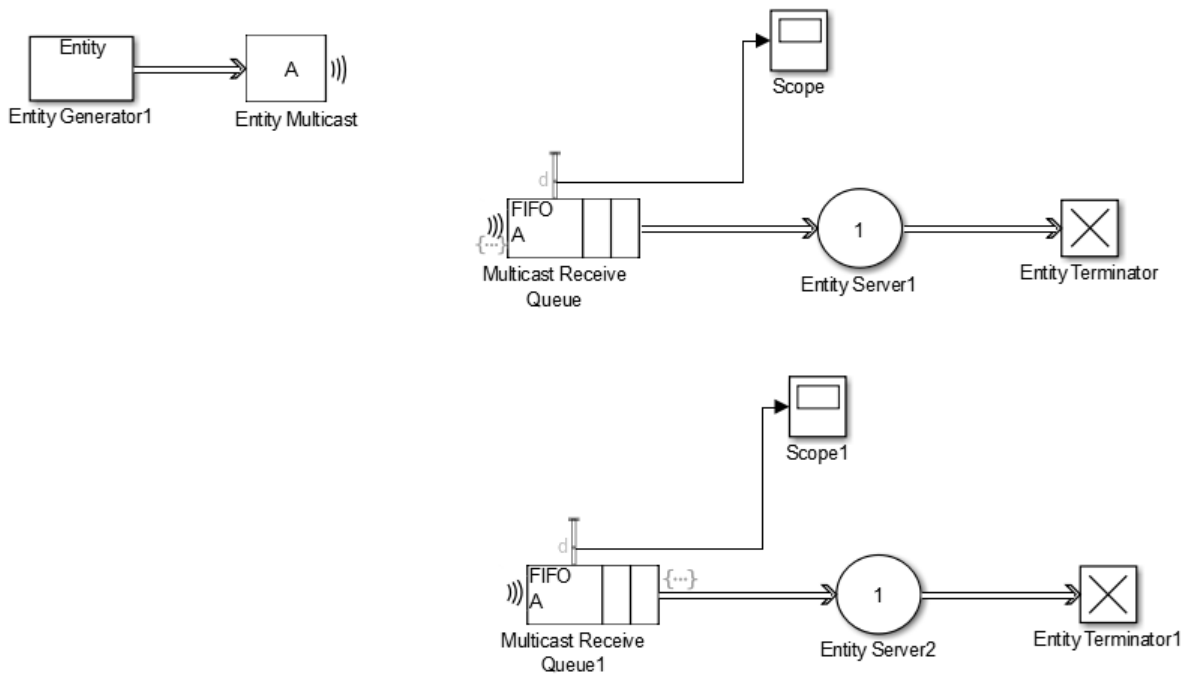
## Entity Resources

Resources are commodities shared by entities in your model. They are independent of entities and attributes, and can exist in the model even if no entity exists or uses them. Resources are different from attributes, which are associated with entities and exist or disappear with their entity.

For example, if you are modeling a restaurant, you can create tables and food as resources for customer entities. Entities can access resources from types of resources. For more information on resources, see "Model with Resources".

## See Also

Entity Multicast | Entity Queue | Entity Server | Multicast Receive Queue | Resource Acquirer | Resource Pool | Resource Releaser

## Related Examples

- "Model Basic Queuing Systems"
- "Model with Resources"

## More About

- "Events and Event Actions"
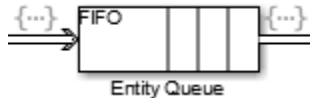
# Write Events Actions

You can write actions for events using MATLAB code or Simulink functions. Each block that enables you to create actions has an **Event Actions** tab. The type of event action you can write depends on the block. For example, for the Entity Queue block you can create event actions for:

· Entity entry to the block

· Exit from the block

· Blocked entities
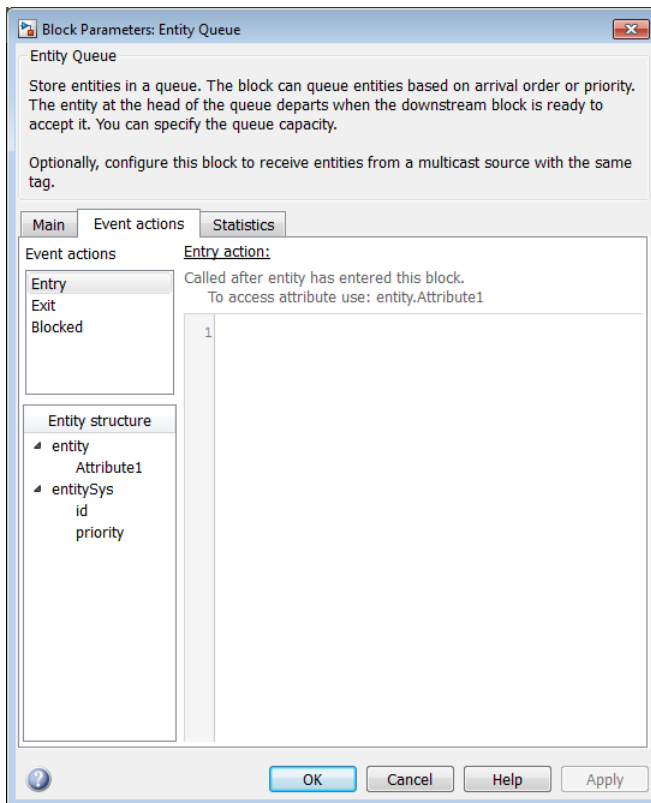
In the actions, entities are available as MATLAB structures, with structure fields representing values of the entity attributes. Reserved fields such as ID and priority are also available as a separate MATLAB structure called entitySys.

When you create an action for the block, a badge appears to indicate that an action exists. One or more badges appear, depending on the action.



Hover over the badge to see what actions exist.
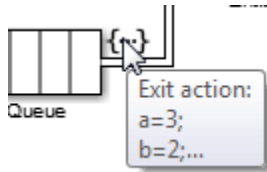


Double-clicking the badge opens the **Event actions** tab of the block.

As you define an action, an asterisk (*) appears in the Event actions tab.

For more information on defining event actions, see "Events and Event Actions".

# See Also
Entity Generator | Entity Queue | Entity Replicator | Entity Server | Entity Terminator | Multicast Receive Queue

## Related Examples
- "Generate Entities When Events Occur"
- "Run Computations on Events"

## More About
- "Events and Event Actions"

# Inspect Statistics

# Statistics Through SimEvents Blocks

The report of statistics is an important part of the SimEvents blocks. Statistics you are most likely to want to see are:

- Server blocks

  - Utilization, average number of entities being served.

- Server and other blocks

  - Number of entities departing the block.
  - Average wait time of entities in the block.

Many SimEvents blocks have a **Statistics** tab, from which you can select the relevant statistic.
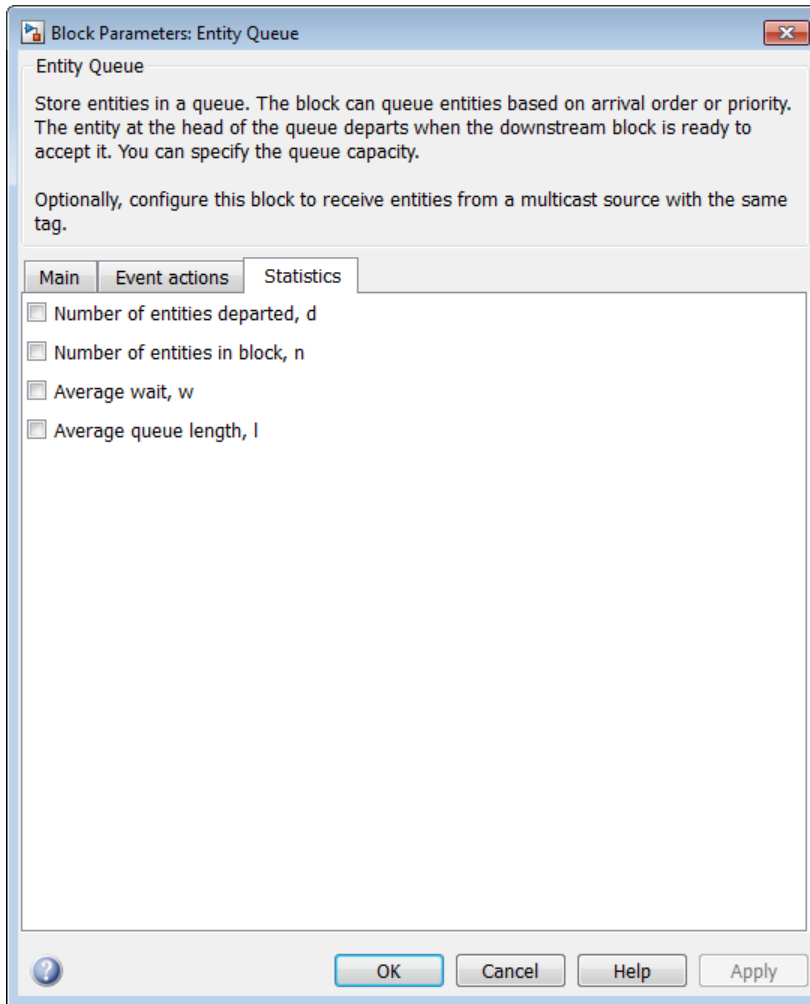
When you request a statistic for a block, the output ports for the block extend from the top of the block, with a label for each port. The Entity Queue can display:

- Number of entities departed
- Number of entities in the block
- Average wait time of the entities

- Average queue length of entities
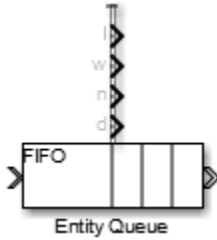


Entity Queue

To display the statistics, connect a display block, such as a Simulink Scope block, to the statistic output port.

## See Also

Entity Queue

## Related Examples

- "Count Entities" on page 4-5
- "Explore Simulation Results Using Plots" on page 2-10
- "Access Statistics from SimEvents Blocks"
- "Count Simultaneous Departures from a Server"

## More About

- "Use Statistics to Understand SimEvents Models"

# Count Entities

| In this section... |
| --- |
| "Count Departures Across the Simulation" on page 4-5 |
| "Count Departures per Time Instant" on page 4-5 |
| "Reset a Counter upon an Event" on page 4-5 |
| "Associate Each Entity with Its Index" on page 4-6 |

Using statistics, you can count entities across the simulation and per time instant.

## Count Departures Across the Simulation

Use the **d** or **a** output from a block to learn how many entities have departed (or arrived at) the block. The output signal also indicates when departures occurred. This method of counting is cumulative throughout the simulation.

## Count Departures per Time Instant

Suppose you want to visualize entity departures from a particular block, and you want to reset (that is, restart) the counter at each time instant. Visualizing departures per time instant can help you:

- Detect simultaneous departures
- Compare the number of simultaneous departures at different time instants
- Visualize the departure times while keeping the plot axis manageable

For an example of counting simultaneous departures from a server, see "Count Simultaneous Departures from a Server".

## Reset a Counter upon an Event

Suppose you want to count entities that depart from a particular block, and you want to reset the counter at arbitrary times during the simulation. Resetting the counter can help you compute statistics for evaluating your system over portions of the simulation.

During the simulation, the block counts departing entities and resets its counter whenever the input signal satisfies your specified event criteria.

## Associate Each Entity with Its Index

To associate an entity with its index, in the initialization section of the Entity Generator block, you can associate an entity with its generation time:

**1**   Use a Simulink Function block with a clock block, such as Digital Clock, to create a Simulink function.

This function returns the current time.

**2**   In the Entity Generator block, create an attribute and associate it with the current time that the Simulink function returns.

For an example, see **Time stamp entities upon generation** in the SimEvents Design Patterns sublibrary.

# See Also

Entity Queue

## Related Examples

·   "Explore Simulation Results Using Plots" on page 2-10
·   "Access Statistics from SimEvents Blocks"
·   "Count Simultaneous Departures from a Server"

## More About

·   "Statistics Through SimEvents Blocks" on page 4-2

# Create Discrete Event Systems Using MATLAB and Stateflow Software

# Custom Discrete Event Systems

The MATLAB Discrete Event System block and Discrete Event Chart block create SimEvents custom blocks.

- MATLAB Discrete Event System — Extends System objects to create custom SimEvents blocks in your model. This capability is useful for including algorithms.
- Discrete Event Chart — Uses Stateflow charts and messages to create custom blocks in your model. This capability is useful for creating application-oriented blocks and add-on software. Examples of such software are real-time operating system schedules and communication networks.

**Note** With just SimEvents and its required software, you can implement your Discrete Event Chart custom block. However, to simulate the model you must have a Stateflow license.

## See Also

Discrete Event Chart | MATLAB Discrete-Event System

### More About

- "Discrete-Event Systems Created with Stateflow Charts"
- "How Discrete-Event Charts Differ from Stateflow Charts"
- "SimEvents Common Design Patterns" on page 1-18
- "System Object Integration" (Simulink)

# Simulate Multidomain Models
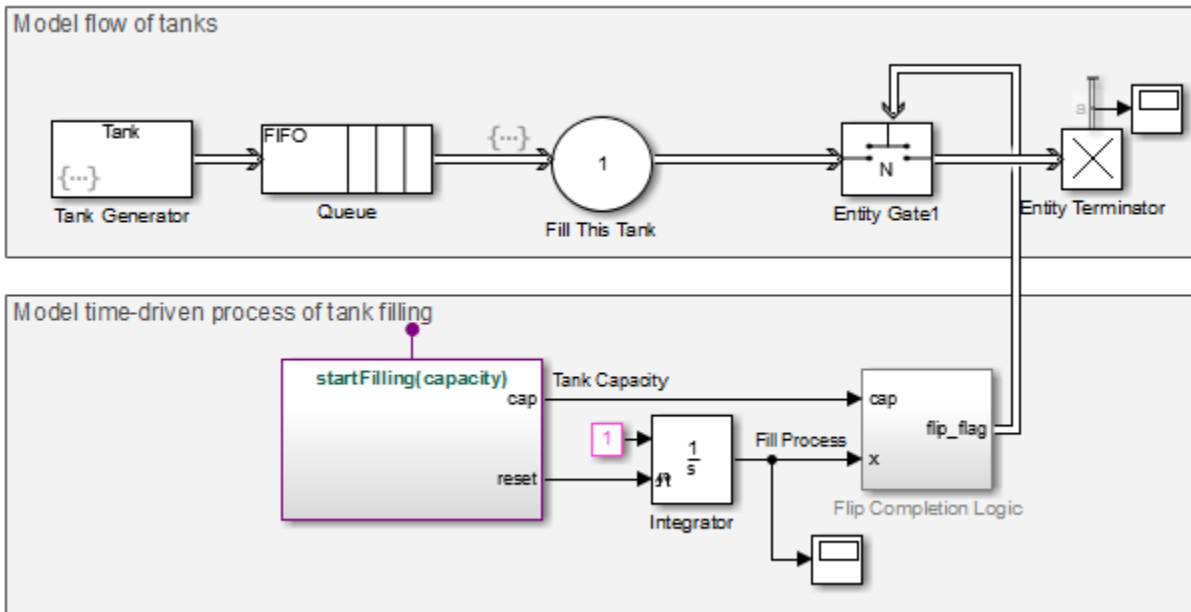
# Simulate a Hybrid System

| In this section... |
|---|
| "SimEvents Part of Model" on page 6-2 |
| "Simulink Part of Model" on page 6-3 |
| "Run the Hybrid Model" on page 6-4 |
| "Event-Based and Time-Based Dynamics in the Simulation" on page 6-5 |

The `seExampleTankFilling` model incorporates both time-based and event-based modeling. It models tanks queuing up to be filled.



The `seExampleTankFilling` example has two sections, a SimEvents part that models event-based behavior and a Simulink part that models continuous-time dynamics.

## SimEvents Part of Model

The SimEvents part models the flow of tanks.

- The Entity Generator block generates the tanks.
- The Entity Queue block queues each tank in FIFO mode.
- The Entity Server block calls the `startFilling` Simulink function to fill each tank.
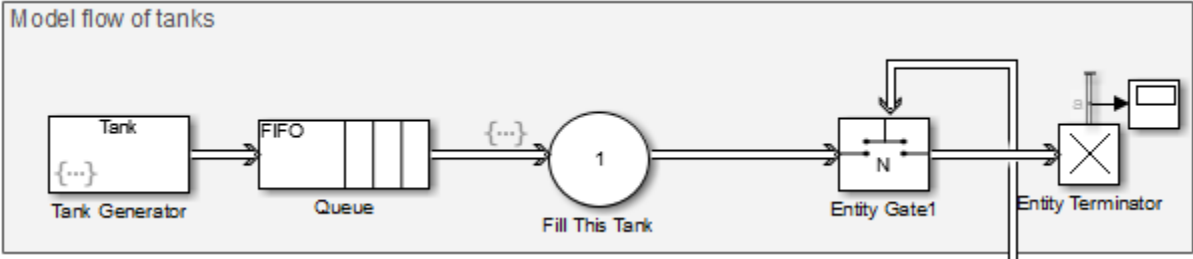


## Simulink Part of Model

The Simulink part models the time-driven process of filling tanks.

- The Simulink side of the model contains the logic to fill the tanks.
- Each tank has a `Capacity` attribute. The continuous time part models the process of filling up a tank, modeled by the Integrator block. When a tank is filled to capacity, the Entity Gate block releases the tank and it departs.
- The Simulink side of the model also contains the Simulink function `startFilling`.
- The Flip Completion Logic subsystem completes the filling of the tank and reinitializes for the next fill. It uses the Entity Gate block to release each tank.
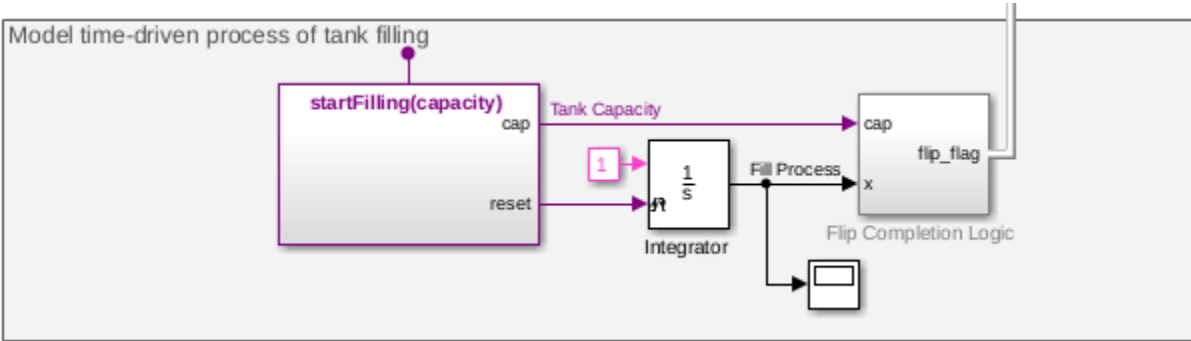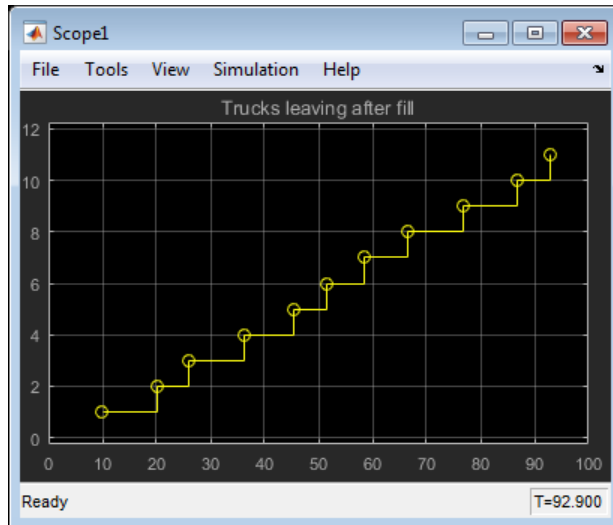
## Run the Hybrid Model

Run the `seExampleTankFilling` model. In the first scope, observe the fill process.
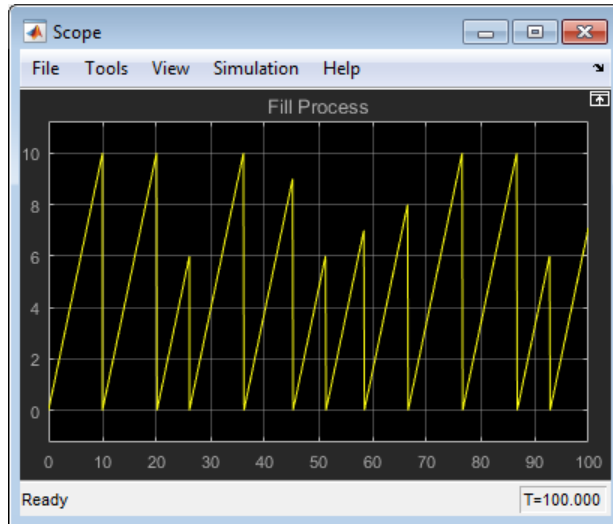


In the second scope, observe the number of trucks leaving after being filled.

## Event-Based and Time-Based Dynamics in the Simulation

In the `seExampleTankFilling` model, the time-based dynamics of the tank fill coexist with the event-based dynamics of the tank flow subsystem. When you run the simulation, the solver and the event calendar both play a role. Upon major time steps of the solver, the simulation solves the ordinary differential equations that represent the dynamics of the tank fill system. Solving the event-based dynamics entails scheduling and processing events, such as service completion and entity generation, on the SimEvents event calendar. Because the model uses a variable-step solver, when events occur in the discrete-event system, the solver has a major time step.

# See Also

Entity Server | Entity Generator | Entity Queue

## More About

- "Discrete-Event Simulation in Simulink Models" on page 1-3
- "Solvers for Discrete-Event Systems"

# 7

# Selected Bibliography

[1] Banks, Jerry, John Carlson, and Barry Nelson. *Discrete-Event System Simulation*, Second Ed. Upper Saddle River, N.J.: Prentice-Hall, 1996.

[2] Cassandras, Christos G. *Discrete Event Systems: Modeling and Performance Analysis*. Homewood, Illinois: Irwin and Aksen Associates, 1993.

[3] Cassandras, Christos G., and Stéphane Lafortune. *Introduction to Discrete Event Systems*. Boston: Kluwer Academic Publishers, 1999.

[4] Fishman, George S. *Discrete-Event Simulation: Modeling, Programming, and Analysis*. New York: Springer-Verlag, 2001.

[5] Gordon, Geoffery. *System Simulation*, Second Ed. Englewood Cliffs, N.J.: Prentice-Hall, 1978.

[6] Kleinrock, Leonard. *Queueing Systems, Volume I: Theory*. New York: Wiley, 1975.

[7] Law, Averill M., and W. David Kelton. *Simulation Modeling and Analysis*, 3rd Ed. New York: McGraw-Hill, 1999.

[8] Moler, C. "Floating points: IEEE Standard unifies arithmetic model," Cleve's Corner. The MathWorks, Inc., 1996. http://www.mathworks.com/company/newsletters/news_notes/pdf/Fall96Cleve.pdf.

[9] Watkins, Kevin. *Discrete Event Simulation in C*. London: McGraw-Hill, 1993.

[10] Zeigler, Bernard P., Herbert Praehofer, and Tag Gon Kim. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Second Ed. San Diego: Academic Press, 2000.