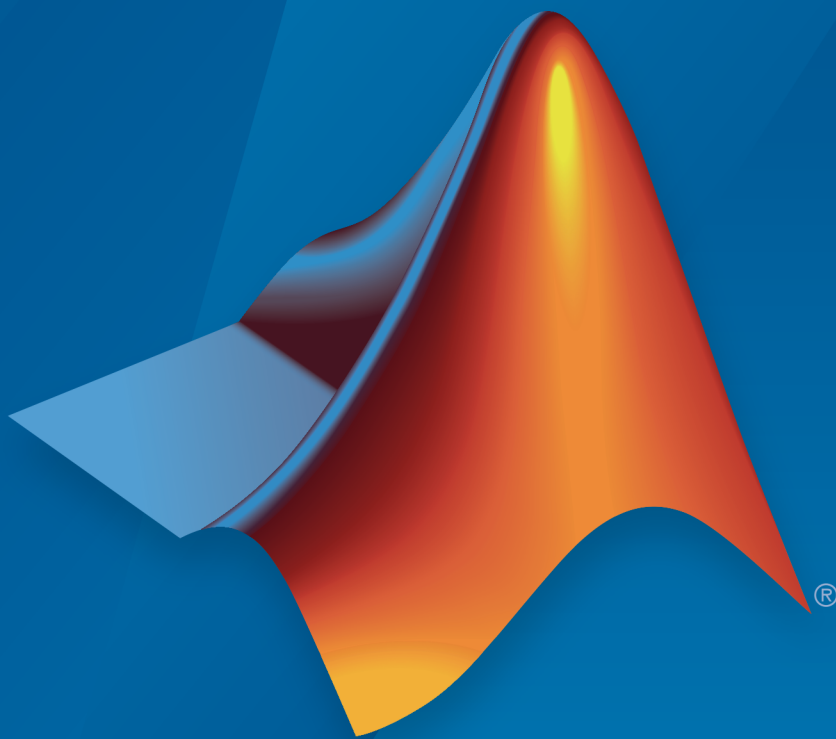


SimEvents<sup>®</sup>

Reference



MATLAB<sup>®</sup>&SIMULINK<sup>®</sup>

R2016a



## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



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

### *SimEvents*<sup>®</sup> Reference

© COPYRIGHT 2005–2016 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

**FEDERAL ACQUISITION:** This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

### **Trademarks**

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### **Patents**

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

## Revision History

March 2007	Online only	Revised for Version 2.0 (Release 2007a). Previously part of <i>SimEvents® User's Guide</i> .
September 2007	Online only	Revised for Version 2.1 (Release 2007b)
March 2008	Online only	Revised for Version 2.2 (Release 2008a)
October 2008	Online only	Revised for Version 2.3 (Release 2008b)
March 2009	Online only	Revised for Version 2.4 (Release 2009a)
September 2009	Online only	Revised for Version 3.0 (Release 2009b)
March 2010	Online only	Revised for Version 3.1 (Release 2010a)
September 2010	Online only	Revised for Version 3.1.1 (Release 2010b)
April 2011	Online only	Revised for Version 3.1.2 (Release 2011a)
September 2011	Online only	Revised for Version 4.0 (Release 2011b)
March 2012	Online only	Revised for Version 4.1 (Release 2012a)
September 2012	Online only	Revised for Version 4.2 (Release 2012b)
March 2013	Online only	Revised for Version 4.3 (Release 2013a)
September 2013	Online only	Revised for Version 4.3.1 (Release 2013b)
March 2014	Online only	Revised for Version 4.3.2 (Release 2014a)
October 2014	Online only	Revised for Version 4.3.3 (Release 2014b)
March 2015	Online only	Revised for Version 4.4 (Release 2015a)
September 2015	Online only	Revised for Version 4.4.1 (Release 2015b)
March 2016	Online only	Revised for Version 5.0 (Release 2016a)



## Alphabetical List

1

## Blocks — Alphabetical List

2

## Configuration Parameters

3

<b>SimEvents Pane</b> .....	3-2
SimEvents Pane Overview .....	3-4
Execution order .....	3-5
Seed for event randomization .....	3-6
Maximum events per block .....	3-7
Maximum events per model .....	3-8
Prevent duplicate events on multiport blocks and branched signals .....	3-8
<b>SimEvents Diagnostics Pane</b> .....	3-10
Diagnostics Pane Overview .....	3-12
Attribute output delayed relative to entities .....	3-13
Response to function call delayed relative to entities .....	3-15
Statistical output delayed relative to entities .....	3-17
Modification of attribute values used for decision making ..	3-19
Identical seeds for random number generators .....	3-21

## Upgrade Advisor Checks

---

# 4

<b>SimEvents Upgrade Advisor Checks</b> .....	4-2
Checks Overview .....	4-2
Check for implicit event duplication caused by SimEvents blocks .....	4-3

## SimEvents Terminology

---

# Alphabetical List

---

# matlab.DiscreteEventSystem class

**Package:** matlab

**Superclasses:** matlab.System

Base class for discrete-event system objects

## Description

matlab.DiscreteEventSystem is the base class for discrete-event System objects. In your class definition file, you must subclass your object from this base class (or from another class that derives from this base class). Subclassing allows you to use the implementation and service methods provided by this base class to build your object. Type this syntax as the first line of your class definition file to directly inherit from the matlab.DiscreteEventSystem base class, where `ObjectName` is the name of your object:

```
classdef ObjectName < matlab.DiscreteEventSystem
```

---

**Note:** You must set `Access = protected` for each `matlab.DiscreteEventSystem` method you use in your code.

---

## Methods

This list contains the methods to implement and the utility methods.

blockedImpl	Event action when entity forward fails
cancelDestroy	Cancel previously scheduled entity destroy event
cancelForward	Cancel previously scheduled forward events
cancelGenerate	Cancel previously scheduled entity generation event
cancelIterate	Cancel previously scheduled iterate event
cancelTimer	Cancel previously scheduled timer event



<code>destroyImpl</code>	Event action upon entity destruction
<code>entityType</code>	Define entity type
<code>entryImpl</code>	Event action when entity enters storage element
<code>eventDestroy</code>	Create entity destroy event
<code>eventForward</code>	Create entity forward event
<code>eventGenerate</code>	Create entity generate event
<code>eventIterate</code>	Create entity iterate event
<code>eventTimer</code>	Create entity timer event
<code>exitImpl</code>	Event action before entity exit from storage
<code>generateImpl</code>	Event action upon entity creation
<code>getEntityPortsImpl</code>	Define input ports and output ports of discrete-event system
<code>getEntityStorageImpl</code>	Define entity storage elements of discrete-event system
<code>getEntityTypesImpl</code>	Define entity types of discrete-event system
<code>iterateImpl</code>	Event action when entity iterates
<code>queueFIFO</code>	Define first-in first-out (FIFO) queue storage
<code>queueLIFO</code>	Define last-in last-out (LIFO) queue storage
<code>queuePriority</code>	Define priority queue storage
<code>queueSysPriority</code>	Define system priority queue storage
<code>setupEventsImpl</code>	Initialize entity generation events
<code>timerImpl</code>	Event action when timer completes

## Inherited Methods

The `matlab.DiscreteEventSystem` class inherits a subset of the `matlab.System` class.

<code>getIconImpl</code>	Name to display as block icon
<code>getHeaderImpl</code>	Header for System object display

<code>getPropertyGroupsImpl</code>	Property groups for System object display
<code>isInactivePropertyImpl</code>	Inactive property status
<code>validatePropertiesImpl</code>	Validate property values
<code>processTunedPropertiesImpl</code>	Action when tunable properties change
<code>getNumInputsImpl</code>	Number of inputs to step method
<code>getInputNamesImpl</code>	Names of System block input ports
<code>getNumOutputsImpl</code>	Number of outputs from step method
<code>getOutputNamesImpl</code>	Names of System block output ports
<code>getOutputSizeImpl</code>	Sizes of output ports
<code>getOutputDataTypeImpl</code>	Data types of output ports
<code>isOutputComplexImpl</code>	Complexity of output ports
<code>getDiscreteStateSpecificationImpl</code>	Discrete state size, data type, and complexity
<code>getDiscreteStateImpl</code>	Discrete state property values
<code>setupImpl</code>	Initialize System object
<code>resetImpl</code>	Reset System object states
<code>releaseImpl</code>	Release resources
<code>loadObjectImpl</code>	Load System object from MAT file
<code>saveObjectImpl</code>	Save System object in MAT file
<code>infoImpl</code>	Information about System object
<code>isDoneImpl</code>	End-of-data flag

## See Also

`matlab.DiscreteEventSystem.cancelDestroy` |  
`matlab.DiscreteEventSystem.cancelGenerate` |  
`matlab.DiscreteEventSystem.cancelIterate` | `matlab.DiscreteEventSystem.cancelTimer`  
| `matlab.DiscreteEventSystem.destroyImpl` | `matlab.DiscreteEventSystem.entityType`  
| `matlab.DiscreteEventSystem.entryImpl` | `matlab.DiscreteEventSystem.eventDestroy`  
| `matlab.DiscreteEventSystem.eventForward` |  
`matlab.DiscreteEventSystem.eventGenerate` |  
`matlab.DiscreteEventSystem.eventIterate` | `matlab.DiscreteEventSystem.eventTimer`  
| `matlab.DiscreteEventSystem.exitImpl` | `matlab.DiscreteEventSystem.generateImpl`  
| `matlab.DiscreteEventSystem.getEntityPortsImpl` |  
`matlab.DiscreteEventSystem.getEntityStorageImpl` |

matlab.DiscreteEventSystem.getEntityTypesImpl |  
matlab.DiscreteEventSystem.iterateImpl | matlab.DiscreteEventSystem.queueFIFO |  
matlab.DiscreteEventSystem.queueLIFO | matlab.DiscreteEventSystem.queuePriority  
| matlab.DiscreteEventSystem.queueSysPriority  
| matlab.DiscreteEventSystem.setupEventsImpl |  
matlab.DiscreteEventSystem.timerImpl

## Related Examples

- Simevents Examples

## More About

- Class Attributes
- Property Attributes

**Introduced in R2016a**

## blockedImpl

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Event action when entity forward fails

### Syntax

```
[entity,events]=blockedImpl(obj,storage,entity,destination)
[entity,events,out1,...]=blockedImpl(obj,storage,entity,destination,
in1,...)
```

### Description

[entity,events]=blockedImpl(obj,storage,entity,destination) specifies event actions of the object when an entity forward fails because the destination storage element has reached its maximum capacity.

[entity,events,out1,...]=blockedImpl(obj,storage,entity,destination,in1,...) specifies such event actions of the object when the block has one or more input signal ports and/or signal output ports.

### Input Arguments

**obj** — Discrete-event System object™

MATLAB® object

Discrete-event System object.

**storage** — Storage

double

Index of the storage element.

**entity** — Entity

MATLAB structure

Entity leaving storage element.

**destination — Destination**

MATLAB structure

Destination of entity, such as an output port or a storage element.

**in1 — Signal inputs**

any value

Any data inputs of the object. These input arguments exist only when the object has data inputs.

## Output Arguments

**entity — Entity**

MATLAB structure

Entity leaving storage, possibly with changed data.

**events — Events**

vector of MATLAB structures

Events to be scheduled after the method returns.

**out1 — Signal output**

any value

Data outputs of the object. You must specify these output arguments when the object has data outputs.

## Examples

**Cancel Current Forward Event**

Cancel the current forward event upon blocking. Schedule an event to forward the entity to the next location. Destroy the entity if no storage can accept the entity.

```
function [entity,events] = blockedImpl(obj,storage,entity,dst)
    % Cancel the current forward event. Schedule an event to
```

```
% forward the entity to the next location. Destroy the entity
% if no storage can accept the entity.
if dst.index < obj.numStorage
    events = [...
        obj.cancelForward(dst.type, dst.index), ...
        obj.eventForward('storage', dst.index+1, 0)];
else
    events = [...
        obj.cancelForward(dst.type, dst.index), ...
        obj.eventDestroy()];
end
end
```

- Simevents Examples

## See Also

[matlab.DiscreteEventSystem](#) | [matlab.DiscreteEventSystem.destroyImpl](#) | [matlab.DiscreteEventSystem.entryImpl](#) | [matlab.DiscreteEventSystem.exitImpl](#) | [matlab.DiscreteEventSystem.generateImpl](#) | [matlab.DiscreteEventSystem.getEntityPortsImpl](#) | [matlab.DiscreteEventSystem.getEntityStorageImpl](#) | [matlab.DiscreteEventSystem.getEntityTypesImpl](#) | [matlab.DiscreteEventSystem.iterateImpl](#) | [matlab.DiscreteEventSystem.setupEventsImpl](#) | [matlab.DiscreteEventSystem.timerImpl](#)

## More About

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**

# cancelDestroy

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Cancel previously scheduled entity destroy event

## Syntax

```
event=cancelDestroy()
```

## Description

`event=cancelDestroy()` cancels a previously scheduled destroy event of the current entity. You can then schedule this event by returning it as the output argument when implementing an event action method, such as `matlab.DiscreteEventSystem.entryImpl` or `matlab.DiscreteEventSystem.exitImpl`.

## Output Arguments

**event** — Event

MATLAB structure

Event for cancelling entity destroy.

## Examples

### Cancel Previously Scheduled Destroy Event

Cancel the previously scheduled destroy event of the entity in the current event action context.

```
function [entity,events] = timerImpl(obj,storage,entity,tag)
    % Cancel the previously scheduled destroy event of the entity in
    % current event action context.
```

```
    event = obj.cancelDestroy();  
end
```

- Simevents Examples

## See Also

matlab.DiscreteEventSystem.cancelForward |  
matlab.DiscreteEventSystem.cancelGenerate |  
matlab.DiscreteEventSystem.cancelIterate | matlab.DiscreteEventSystem.cancelTimer  
| matlab.DiscreteEventSystem.eventDestroy |  
matlab.DiscreteEventSystem.eventForward |  
matlab.DiscreteEventSystem.eventGenerate | matlab.DiscreteEventSystem.eventIterate  
| matlab.DiscreteEventSystem.eventTimer

## More About

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**



# cancelForward

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Cancel previously scheduled forward events

## Syntax

```
event=cancelForward(destinationType,destinationID)
```

## Description

`event=cancelForward(destinationType,destinationID)` cancel previously scheduled forward events on the current entity. You can then schedule this event by returning it as the output argument when implementing an event action method, such as `matlab.DiscreteEventSystem.entryImpl` or `matlab.DiscreteEventSystem.exitImpl`.

## Input Arguments

**destinationType** — Destination type

string

Destination type, specified as a string. Its value can be either:

- `storage`, if destination of the forward event is a storage element.
- `output`, if destination of forward event is an output port.

**destinationID** — Destination index

double

Destination index, specified as a double. Its value can be either:

- Storage index, when `destinationType` is `storage`.
- Output port index, when `destinationType` is `output`.

## Output Arguments

### **event** — Event

MATLAB structure

Event for cancelling an entity forward.

## Examples

### Cancel Previously Schedule Forward Event

Cancel a previously scheduled forward event of the entity in the current event action context.

```
function [entity,events] = timerImpl(obj,storage,entity,tag)
    % Cancel a previously scheduled forward event of the entity in
    % current event action context. The entity was scheduled to go to
    % storage element 2.
    event1 = obj.cancelForward('storage', 2);

    % Cancel a previously scheduled forward event of the entity in
    % current event action context. The entity was scheduled to go to
    % output port 1.
    event2 = obj.cancelForward('output', 1);
end
```

- Simevents Examples

### See Also

matlab.DiscreteEventSystem.cancelDestroy |  
matlab.DiscreteEventSystem.cancelGenerate |  
matlab.DiscreteEventSystem.cancelIterate | matlab.DiscreteEventSystem.cancelTimer  
| matlab.DiscreteEventSystem.eventDestroy |  
matlab.DiscreteEventSystem.eventForward |  
matlab.DiscreteEventSystem.eventGenerate | matlab.DiscreteEventSystem.eventIterate  
| matlab.DiscreteEventSystem.eventTimer

### More About

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**

## cancelGenerate

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Cancel previously scheduled entity generation event

## Syntax

```
event=cancelGenerate(storageID,tag)
```

## Description

`event=cancelGenerate(storageID,tag)` cancels a previously scheduled generation event. You can then schedule this event by returning it as the output argument when implementing an event action method, such as `matlab.DiscreteEventSystem.entryImpl` or `matlab.DiscreteEventSystem.exitImpl`.

## Input Arguments

**storageID** — Storage index

double

Storage index of the to-be-cancelled entity generation event.

**tag** — Tag

string

Tag of the to-be-cancelled entity generation event.

## Output Arguments

**event** — Event

MATLAB structure

Event for cancelling an entity generation.

## Examples

### Cancel Previously Scheduled Entity Generation Event

Cancel a previously scheduled entity generation event.

```
function [entity,event] = entryImpl(obj,storage,entity,src)
    % Cancel a previously scheduled entity generation event. The event
    % was scheduled for storage element 3, with a custom tag 'seed'.
    event = obj.cancelGenerate(3, 'seed');
end
```

- [Simevents Examples](#)

### See Also

[matlab.DiscreteEventSystem.cancelDestroy](#) |  
[matlab.DiscreteEventSystem.cancelGenerate](#) |  
[matlab.DiscreteEventSystem.cancelIterate](#) | [matlab.DiscreteEventSystem.cancelTimer](#)  
| [matlab.DiscreteEventSystem.eventDestroy](#) |  
[matlab.DiscreteEventSystem.eventForward](#) |  
[matlab.DiscreteEventSystem.eventGenerate](#)

### More About

- [“Discrete-Event Systems Created with a System Object”](#)

**Introduced in R2016a**

## cancelIterate

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Cancel previously scheduled iterate event

### Syntax

```
event=cancelIterate(storageID,tag)
```

### Description

`event=cancelIterate(storageID,tag)` cancels a previously scheduled iterate event. You can commit the cancellation by returning it as the output argument when implementing an event action method, such as `matlab.DiscreteEventSystem.entryImpl`.

### Input Arguments

**storageID** — Storage index

double

Storage index of the to-be-cancelled iterate event.

**tag** — Tag

string

Tag of the to-be-cancelled iterate event.

### Output Arguments

**event** — Event

MATLAB structure

Event for cancelling the specified iterate event.

## Examples

### Cancel Previously Scheduled Iterate Event

Cancel a previously scheduled iterate event.

```
function [entity,event] = entryImpl(obj,storage,entity,src)
    % Cancel a previously scheduled iterate event. The event was
    % scheduled for storage element 2, with a custom tag 'search'.
    event = obj.cancelIterate(2, 'search');
end
```

- [Simevents Examples](#)

### See Also

[matlab.DiscreteEventSystem.cancelDestroy](#) |  
[matlab.DiscreteEventSystem.cancelForward](#) |  
[matlab.DiscreteEventSystem.cancelGenerate](#) |  
[matlab.DiscreteEventSystem.cancelTimer](#) | [matlab.DiscreteEventSystem.eventDestroy](#)  
| [matlab.DiscreteEventSystem.eventForward](#) |  
[matlab.DiscreteEventSystem.eventGenerate](#) | [matlab.DiscreteEventSystem.eventIterate](#)  
| [matlab.DiscreteEventSystem.eventTimer](#)

### More About

- [“Discrete-Event Systems Created with a System Object”](#)

**Introduced in R2016a**

## cancelTimer

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Cancel previously scheduled timer event

### Syntax

```
event=cancelTimer(tag)
```

### Description

`event=cancelTimer(tag)` cancels a previously scheduled timer event of the current entity. You can commit the cancellation by returning it as the output argument when implementing an event action method, such as `matlab.DiscreteEventSystem.entryImpl`.

### Input Arguments

**tag** — Tag

string

Tag of the to-be-cancelled timer event.

### Output Arguments

**event** — Event

MATLAB structure

Event for cancelling the specified timer.



## Examples

### Cancel Previously Scheduled Timer Event

Cancel a previously scheduled timer event of the entity in the current event action context.

```
function [entity,event] = entryImpl(obj,storage,entity,src)
    % Cancel a previously scheduled timer event of the entity in
    % current event action context. The event was scheduled with a
    % custom tag 'timeout'.
    event = obj.cancelTimer('timeout');
end
```

- [Simevents Examples](#)

### See Also

[matlab.DiscreteEventSystem.cancelDestroy](#) |  
[matlab.DiscreteEventSystem.cancelForward](#) |  
[matlab.DiscreteEventSystem.cancelGenerate](#) |  
[matlab.DiscreteEventSystem.cancelIterate](#) | [matlab.DiscreteEventSystem.eventDestroy](#)  
| [matlab.DiscreteEventSystem.eventForward](#) |  
[matlab.DiscreteEventSystem.eventGenerate](#) | [matlab.DiscreteEventSystem.eventIterate](#)  
| [matlab.DiscreteEventSystem.eventTimer](#)

### More About

- [“Discrete-Event Systems Created with a System Object”](#)

**Introduced in R2016a**

## destroyImpl

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Event action upon entity destruction

### Syntax

```
[events]=destroyImpl(obj,storage,entity)
```

```
[events,out1,...]=destroyImpl(obj,storage,entity,in1,...)
```

### Description

[events]=destroyImpl(obj,storage,entity) specifies event actions of the object before an entity is destroyed.

[events,out1,...]=destroyImpl(obj,storage,entity,in1,...) specifies such event actions of the object when the block has one or more input signal ports and/or signal output ports.

### Input Arguments

**obj** — Discrete-event System object

MATLAB object

Discrete-event System object.

**storage** — Storage

double

Index of the storage element.

**entity** — Entity

MATLAB structure

Entity leaving storage element.

**in1 — Signal input**

any value

Any data inputs of the object. These input arguments exist only when the object has data inputs.

## Output Arguments

**events — Events**

vector of MATLAB structures

Events to be scheduled.

**out1 — Signal output**

any value

Data outputs of the object. You must specify these output arguments when the object has data outputs.

## Examples

**Event Action Upon Entity Destruction**

Specify event action upon entity destruction in storage.

```
function events = destroyImpl(obj,storage,entity)
    % Upon destroy of an entity, display its ID and schedule to
    % generate a new entity.
    disp(['Entity of ID ' num2str(entity.sys.id) ' is destroyed']);
    events = obj.eventGenerate(storage, 'Refill', 1, entity.sys.priority);
end
```

- Simevents Examples

**See Also**

matlab.DiscreteEventSystem | matlab.DiscreteEventSystem.blockedImpl |  
 matlab.DiscreteEventSystem.entryImpl | matlab.DiscreteEventSystem.exitImpl  
 | matlab.DiscreteEventSystem.generateImpl |

matlab.DiscreteEventSystem.getEntityPortsImpl |  
matlab.DiscreteEventSystem.getEntityStorageImpl |  
matlab.DiscreteEventSystem.getEntityTypesImpl |  
matlab.DiscreteEventSystem.iterateImpl |  
matlab.DiscreteEventSystem.setupEventsImpl |  
matlab.DiscreteEventSystem.timerImpl

## **More About**

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**

# entityType

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Define entity type

## Syntax

```
entitytype=entityType(name)
```

```
entitytype=entityType(name,datatype)
```

```
entitytype=entityType(name,datatype,dimensions,complexity)
```

## Description

`entitytype=entityType(name)` defines a named entity type.

`entitytype=entityType(name,datatype)` defines a named entity type with a specified data type.

`entitytype=entityType(name,datatype,dimensions,complexity)` defines a named entity type with a specified data type, dimensions, and complexity.

## Input Arguments

**name** — Entity type name

string

Entity type name, specified as a string.

**datatype** — Data type

string

Data type, specified as a string, that specifies the data type of the entity. The data type must be a built-in data type or a bus object.

**dimensions** — Dimensions

vector of doubles

Dimensions, specified as a vector of doubles, specifying the dimensions of the entity.

## **complexity** — Complexity

logical | double

Complexity, specified as a logical or double value, specifying the complexity of the entity:

- `false` or `0` — If the entity contains real values.
- `true` or any positive number — If the entity contains complex values.

## Output Arguments

### **entitytype** — Entity type

MATLAB structure

Entity type, specified as a MATLAB structure.

## Examples

### Define Entity Type

Define entity types `type1`, `type2`, and `type3`.

```
function entityTypes = getEntityTypesImpl(obj)
    % Define entity type 'type1' with inherited data type, dimension
    % and complexity
    t1 = obj.entityType('type1');

    % Define entity type 'type2' with specified data type ('mybus'),
    % default dimension and complexity (i.e. scalar real values)
    t2 = obj.entityType('type2', 'mybus');

    % Define entity type 'type3' with specified data type ('double'),
    % dimension (2 by 3 matrix), and complexity (complex)
    t3 = obj.entityType('type3', 'double', [2 3], true);

    entityTypes = [t1, t2, t3];
end
```

- Simevents Examples

## See Also

matlab.DiscreteEventSystem | matlab.DiscreteEventSystem.getEntityTypesImpl

## More About

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**

## entryImpl

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Event action when entity enters storage element

## Syntax

```
[entity,events]=entryImpl(obj,storage,entity,source)
[entity,events,out1,...]=entryImpl(obj,storage,entity,source,
in1,...)
```

## Description

[entity,events]=entryImpl(obj,storage,entity,source) specifies event actions of the object when an entity enters a storage.

[entity,events,out1,...]=entryImpl(obj,storage,entity,source,in1,...) such event actions of the object when the block has one or more input signal ports and/or signal output ports.

## Input Arguments

**obj** — Discrete-event System object

MATLAB object

Discrete-event System object.

**storage** — Storage

double

Index of the storage element.

**entity** — Entity

MATLAB structure



Entity entering storage component.

**source — Source location**

MATLAB structure

Source location of entity, such as an input port or a storage element.

**in1 — Signal input**

any value

Any data inputs of the object. These input arguments exist only when the object has data inputs.

## Output Arguments

**entity — Entity**

MATLAB structure

Entity entering storage, possibly with changed data.

**events — Events**

vector of MATLAB structures

Events to be scheduled.

**out1 — Signal output**

any value

Data outputs of the object. You must specify these output arguments when the object has data outputs.

## Examples

**Event Action Upon Entity Entry**

Event action for entity entry to storage.

```
function [entity,events] = entryImpl(obj,storage,entity,src)
    % Specify event actions when entity entered storage.
    disp(['Entity of ID ' num2str(entity.sys.id) ...
```

```
        ' has entered storage element ' num2str(storage));
switch src.type
case 'input'
    disp(['Entity came from input port ' num2str(src.index)]);
case 'storage'
    disp(['Entity came from storage element ' num2str(src,index)]);
end
events = [ ...
    obj.eventDestroy(), ...    % Destroy the newly entered entity
    obj.eventIterate(2, '')]; % Iterate entities in storage element 2
end
```

- Simevents Examples

## See Also

[matlab.DiscreteEventSystem](#) | [matlab.DiscreteEventSystem.blockedImpl](#) |  
[matlab.DiscreteEventSystem.destroyImpl](#) | [matlab.DiscreteEventSystem.entryImpl](#)  
| [matlab.DiscreteEventSystem.exitImpl](#) | [matlab.DiscreteEventSystem.generateImpl](#)  
| [matlab.DiscreteEventSystem.getEntityPortsImpl](#) |  
[matlab.DiscreteEventSystem.getEntityStorageImpl](#) |  
[matlab.DiscreteEventSystem.getEntityTypesImpl](#) |  
[matlab.DiscreteEventSystem.iterateImpl](#) |  
[matlab.DiscreteEventSystem.setupEventsImpl](#) |  
[matlab.DiscreteEventSystem.timerImpl](#)

## More About

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**

# eventDestroy

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Create entity destroy event

## Syntax

```
event=eventDestroy()
```

## Description

`event=eventDestroy()` creates an event to destroy an entity. You can then schedule this event by returning it as an output argument when implementing an event action method, such as `matlab.DiscreteEventSystem.timerImpl`.

## Output Arguments

**event** — Event

MATLAB structure

Event that destroys the entity in current event action context.

## Examples

### Destroy Entity in Current Event Action Context

Define an event to destroy the entity in current event action context.

```
function [entity,event] = entryImpl(obj,storage,entity,src)
    % Define an event to destroy the entity in current event action
    % context.
    event = obj.eventDestroy();
end
```

- Simevents Examples

## **See Also**

matlab.DiscreteEventSystem.cancelDestroy |  
matlab.DiscreteEventSystem.cancelForward |  
matlab.DiscreteEventSystem.cancelGenerate |  
matlab.DiscreteEventSystem.cancelIterate | matlab.DiscreteEventSystem.cancelTimer  
| matlab.DiscreteEventSystem.eventForward |  
matlab.DiscreteEventSystem.eventGenerate | matlab.DiscreteEventSystem.eventIterate  
| matlab.DiscreteEventSystem.eventTimer

## **More About**

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**

# eventForward

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Create entity forward event

## Syntax

```
event=eventForward(locationType,locationIndex,delay)
```

## Description

`event=eventForward(locationType,locationIndex,delay)` creates an event to forward an entity from the current location to a new location. You can then schedule this event by returning it as the output argument when implementing an event action method, such as `matlab.DiscreteEventSystem.entryImpl`.

## Input Arguments

### **locationType** — Location type

string

Type of the new location. Specify `'storage'` if the new location is a storage element of the discrete-event system. Specify `'output'` if you want the entity to exit from an output port of the discrete-event system.

### **locationIndex** — Location index

double

Index of the new location. If location type is `'storage'`, it indicates the index of a storage element. If location type is `'output'`, it indicates the index of an output port.

### **delay** — Delay

double

Time delay between current simulation time and the time the entity will be forwarded.

## Output Arguments

### **event** – Event

MATLAB structure

Event that forwards the entity in current event action context to a new location.

## Examples

### **Forward Current Entity to Storage**

Define an event that forwards the current entity to storage.

```
function [entity,events] = entryImpl(obj,storage,entity,src)
    % Define an event that forwards the current entity to storage
    % element 2. Event shall be scheduled to execute 0.8 second later.
    event1 = obj.eventForward('storage', 2, 0.8);

    % Define an event that forwards the current entity to output port 1.
    % Event shall be scheduled to execute at current simulation clock time.
    event2 = obj.eventForward('output', 1, 0);
end
```

- [Simevents Examples](#)

### **See Also**

[matlab.DiscreteEventSystem.cancelDestroy](#) |  
[matlab.DiscreteEventSystem.cancelForward](#) |  
[matlab.DiscreteEventSystem.cancelGenerate](#) |  
[matlab.DiscreteEventSystem.cancelIterate](#) | [matlab.DiscreteEventSystem.cancelTimer](#)  
| [matlab.DiscreteEventSystem.eventDestroy](#) |  
[matlab.DiscreteEventSystem.eventGenerate](#) | [matlab.DiscreteEventSystem.eventIterate](#)  
| [matlab.DiscreteEventSystem.eventTimer](#)

### **More About**

- [“Discrete-Event Systems Created with a System Object”](#)

**Introduced in R2016a**

# eventGenerate

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Create entity generate event

## Syntax

```
event=eventGenerate(storageID,tag,delay,priority)
```

## Description

`event=eventGenerate(storageID,tag,delay,priority)` creates an event to generate an entity. You can then schedule this event by returning it as the output argument when implementing an event action method, such as `matlab.DiscreteEventSystem.entryImpl`.

## Input Arguments

**storageID** — Storage index

double

Index of the storage element, where a new entity will be generated.

**tag** — Tag

string

Custom tag of this entity generate event.

**delay** — Delay

double

Time delay between current simulation time and the time the entity will be generated.

**priority** — Priority

double

Positive integer value indicating system priority of the new entity. A smaller numeric value indicates a higher priority.

## Output Arguments

### **event** — Event

MATLAB structure

Event that generates an new entity in the specified storage element.

## Examples

### Define Entity Generation Event

Define entity generation event in storage element 3.

```
function event = setupEventsImpl(obj)
    % Define an entity generation event
    % - A new entity shall be created in storage element 3
    % - The event has a custom tag 'seed'
    % - The event shall be executed 0.5 second later
    % - The new entity shall be initialized with a priority of 200
    event = obj.eventGenerate(3, 'seed', 0.5, 200);
end
```

- [Simevents Examples](#)

### See Also

[matlab.DiscreteEventSystem.cancelDestroy](#) |  
[matlab.DiscreteEventSystem.cancelForward](#) |  
[matlab.DiscreteEventSystem.cancelGenerate](#) |  
[matlab.DiscreteEventSystem.cancelIterate](#) | [matlab.DiscreteEventSystem.cancelTimer](#)  
| [matlab.DiscreteEventSystem.eventDestroy](#) |  
[matlab.DiscreteEventSystem.eventForward](#) | [matlab.DiscreteEventSystem.eventIterate](#)  
| [matlab.DiscreteEventSystem.eventTimer](#)

### More About

- [“Discrete-Event Systems Created with a System Object”](#)



**Introduced in R2016a**

## eventIterate

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Create entity iterate event

## Syntax

```
event=eventIterate(storageID,tag,priority)
```

## Description

`event=eventIterate(storageID,tag,priority)` creates an event to repeatedly process entities of a storage element. You can then schedule this event by returning it as the output argument when implementing an event action method, such as `matlab.DiscreteEventSystem.exitImpl`.

## Input Arguments

**storageID** — Storage index

double

Index of a storage element. Entities inside this storage element will be iterated.

**tag** — Tag

string

Custom tag of this entity iterate event.

**priority** — Priority

double

Priority of the entity iterate event. This value must be a positive integer, where a smaller value indicates a higher event priority.

## Output Arguments

### event — Event

MATLAB structure

Event that processes entities of a specific storage element.

## Examples

### Iterate Entities of a Storage Element

Define an event to iterate entities of a storage element..

```
function event = exitImpl(obj,storage,entity,dst)
    % Define an event to iterate entities of a storage element
    % - The event is regarding to storage element 2
    % - The event has a custom tag 'search'
    % - The event shall be executed at current simulation clock time
    % - The event has a priority of 10 (a smaller numeric value
    %   indicates a higher event priority)
    event = obj.eventIterate(2, 'search', 10);
end
```

- Simevents Examples

### See Also

matlab.DiscreteEventSystem.cancelDestroy |  
 matlab.DiscreteEventSystem.cancelForward |  
 matlab.DiscreteEventSystem.cancelGenerate |  
 matlab.DiscreteEventSystem.cancelIterate | matlab.DiscreteEventSystem.cancelTimer  
 | matlab.DiscreteEventSystem.eventDestroy |  
 matlab.DiscreteEventSystem.eventForward |  
 matlab.DiscreteEventSystem.eventGenerate | matlab.DiscreteEventSystem.eventTimer

### More About

- “Discrete-Event Systems Created with a System Object”

Introduced in R2016a

## eventTimer

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Create entity timer event

## Syntax

```
event=eventTimer(tag, delay)
```

## Description

`event=eventTimer(tag, delay)` creates an event to delay an entity for a period of time. You can then schedule the timer by returning it as the output argument when implementing an event action method, such as `matlab.DiscreteEventSystem.entryImpl`.

## Input Arguments

**tag** — Tag

string

Custom tag of this entity timer event.

**delay** — Delay

double

Time delay between current simulation time and the time that this timer event will be executed.

## Output Arguments

**event** — Event

MATLAB structure

Event that delays the entity in current event action context for a period of time.

## Examples

### Define Timer Event

Define a timer event.

```
function [entity,event] = entryImpl(obj,storage,entity,src)
    % Define a timer event
    % - The event is regarding the entity in current event action context
    % - The event has a custom tag 'timeout'
    % - The event will be executed 3.0 seconds later
    event = obj.eventTimer('timeout', 3.0);
end
```

- [Simevents Examples](#)

### See Also

[matlab.DiscreteEventSystem.cancelDestroy](#) |  
[matlab.DiscreteEventSystem.cancelForward](#) |  
[matlab.DiscreteEventSystem.cancelGenerate](#) |  
[matlab.DiscreteEventSystem.cancelIterate](#) | [matlab.DiscreteEventSystem.cancelTimer](#)  
| [matlab.DiscreteEventSystem.eventDestroy](#) |  
[matlab.DiscreteEventSystem.eventForward](#) |  
[matlab.DiscreteEventSystem.eventGenerate](#) | [matlab.DiscreteEventSystem.eventIterate](#)

### More About

- [“Discrete-Event Systems Created with a System Object”](#)

**Introduced in R2016a**

## exitImpl

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Event action before entity exit from storage

## Syntax

[events]=exitImpl(obj,storage,entity,destination)

[events,out1,...]=exitImpl(obj,storage,entity,destination,in1,...)

## Description

[events]=exitImpl(obj,storage,entity,destination) specifies event actions of the object when an entity exits a storage.

[events,out1,...]=exitImpl(obj,storage,entity,destination,in1,...) specifies such event actions of the object when the block has one or more input signal ports and/or signal output ports.

## Input Arguments

**obj** — Discrete-event System object

MATLAB object

Discrete-event System object.

**storage** — Storage

double

Index of the storage element.

**entity** — Entity

MATLAB structure

Entity leaving storage element.

**destination — Destination**

MATLAB structure

Destination of entity, such as an output port or a storage element.

**in1 — Data inputs**

any value

Any data inputs of the object. These input arguments exist only when the object has data inputs.

## Output Arguments

**events — Events**

vector of MATLAB structures

Events to be scheduled after the method returns.

**out1 — Signal output**

any value

Data outputs of the object. You must specify these output arguments when the object has data outputs.

## Examples

**Refill Upon Entity Exit Storage**

Create a new entity when an existing entity exits the storage element.

```
function events = exitImpl(obj,storage,entity,dst)
    % Upon exit of an entity, display its ID and schedule to
    % generate a new entity.
    disp(['Entity of ID ' num2str(entity.sys.id) ' has exited']);
    events = obj.eventGenerate(storage, 'Refill', 1, entity.sys.priority);
end
```

- Simevents Examples

## See Also

matlab.DiscreteEventSystem | matlab.DiscreteEventSystem.blockedImpl |  
matlab.DiscreteEventSystem.destroyImpl | matlab.DiscreteEventSystem.entryImpl  
| matlab.DiscreteEventSystem.exitImpl | matlab.DiscreteEventSystem.generateImpl  
| matlab.DiscreteEventSystem.getEntityPortsImpl |  
matlab.DiscreteEventSystem.getEntityStorageImpl |  
matlab.DiscreteEventSystem.getEntityTypesImpl |  
matlab.DiscreteEventSystem.iterateImpl |  
matlab.DiscreteEventSystem.setupEventsImpl |  
matlab.DiscreteEventSystem.timerImpl

## More About

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**



# generateImpl

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Event action upon entity creation

## Syntax

```
[entity,events]=generateImpl(obj,storage,entity,tag)
[entity,events,out1,...]=generateImpl(obj,storage,entity,tag,
in1,...)
```

## Description

[entity,events]=generateImpl(obj,storage,entity,tag) specifies event actions of the object when an entity is created inside a storage component.

[entity,events,out1,...]=generateImpl(obj,storage,entity,tag,in1,...) specifies such event actions of the object when the block has one or more input signal ports and/or signal output ports.

## Input Arguments

**obj** — Discrete-event System object

MATLAB object

Discrete-event System object.

**storage** — Storage

double

Index of the storage element.

**entity** — Entity

MATLAB structure

Entity to create inside storage element.

**tag — Tag**

string

Tag of the current entity generation event.

**in1 — Input**

any value

Any data inputs of the object. These input arguments exist only when the object has data inputs.

## Output Arguments

**entity — Entity**

MATLAB structure

Entities created with possibly changed values.

**events — Events**

vector of MATLAB structures

Events to be scheduled for just after entities are created.

**out1 — Data output**

any value

Data outputs of the object. You must specify these output arguments when the object has data outputs.

## Examples

### Set Initial Values When Entity is Generated

Initialize attribute values when entity is generated in a storage element.

```
function [entity,events] = generateImpl(obj,storage,entity,tag)
    % Specify event actions when entity generated in storage.
    % - For entitiy generation event of tag 'Adam', initialize the
```

```

% entity so that its attribute 'gender' has value '0', and its
% priority is '200'.
% - For entity generation event of tag 'Eve', initialize the
% entity so that its attribute 'gender' has value '1', and its
% priority is '100'.
% - An event is returned to forward the entity to storage
% element 2 with a time delay of 0.6.
switch tag
    case 'Adam'
        entity.data.gender = 0;
        entity.sys.priority = 200;
    case 'Eve'
        entity.data.gender = 1;
        entity.sys.priority = 100;
end
events = obj.eventForward('storage',2,0.6);
end

```

- Simevents Examples

## See Also

[matlab.DiscreteEventSystem.blockedImpl](#) |
[matlab.DiscreteEventSystem.destroyImpl](#) |
[matlab.DiscreteEventSystem.entryImpl](#) |
[matlab.DiscreteEventSystem.exitImpl](#) |
[matlab.DiscreteEventSystem.generateImpl](#) |
[matlab.DiscreteEventSystem.getEntityPortsImpl](#) |
[matlab.DiscreteEventSystem.getEntityStorageImpl](#) |
[matlab.DiscreteEventSystem.getEntityTypesImpl](#) |
[matlab.DiscreteEventSystem.iterateImpl](#) |
[matlab.DiscreteEventSystem.setupEventsImpl](#) |
[matlab.DiscreteEventSystem.timerImpl](#)

## More About

- “Discrete-Event Systems Created with a System Object”

Introduced in R2016a

## getEntityPortsImpl

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Define input ports and output ports of discrete-event system

### Syntax

```
[inputTypes,outputTypes]=getEntityPortsImpl(obj)
```

### Description

[inputTypes,outputTypes]=getEntityPortsImpl(obj) defines input ports and output ports of a discrete-event system.

### Input Arguments

**obj** — Discrete-event System object

MATLAB object

Discrete-event System object.

### Output Arguments

**inputTypes** — Input types

cell vector of strings

Input port types of a discrete-event system, specified as a cell vector of strings with a length that is the same as the number of input ports.

The *N*th element of the vector is a string that specifies the type of the *N*th input port.

- If the port is an entity port, the string indicates the entity type name of this port. The name must match one of the entity types specified in `matlab.DiscreteEventSystem.getEntityTypesImpl`.

- If the port is a signal port, the string must be empty ('').

### **outputTypes — Output types**

cell vector of strings

Output port types of a discrete-event system, specified as a cell vector with a length that is the same as the number of output ports.

The *N*th element of the vector is a string that specifies type of the *N*th output port.

- If the port is an entity port, the string indicates the entity type name of this port. The name must match one of the entity types specified in `matlab.DiscreteEventSystem.getEntityTypesImpl`.
- If the port is a signal port, the string must be empty ('').

## Examples

### Get Entity Inputs and Outputs for Discrete-Event System

Get entity input and output port types for discrete-event system.

```
function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
    % Specify input and output port types.
    %
    % This implementation further specifies port type and entity
    % type at these inputs and outputs:
    % Inputs:
    % 1. Signal port
    % 2. Entity port receiving entities of type 'entity1'
    % 3. Entity port receiving entities of type 'entity2'
    % Outputs:
    % 1. Signal port
    % 2. Entity port sending entities of type 'entity2'
    %
    % The discrete-event system must have already defined:
    % - 3 inputs (by method 'getNumInputsImpl') and
    % - 2 outputs (by method 'getNumOutputsImpl')
    inputTypes = {'', 'entity1', 'entity2'};
    outputTypes = {'', 'entity2'};
end
```

- Simevents Examples

## **See Also**

matlab.DiscreteEventSystem | matlab.DiscreteEventSystem.getEntityStorageImpl |  
matlab.DiscreteEventSystem.getEntityTypesImpl

## **More About**

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**

# getEntityStorageImpl

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Define entity storage elements of discrete-event system

## Syntax

```
[storageSpecs,I]=getEntityStorageImpl(obj)
```

## Description

[storageSpecs,I]=getEntityStorageImpl(obj) defines entity storage elements of a discrete-event system.

## Input Arguments

**obj** — Discrete-event System object

MATLAB object

Discrete-event System object.

## Output Arguments

**storageSpecs** — Storage specifications

vector of MATLAB structures

Entity storage specifications of a discrete-event system, specified as a vector of MATLAB structures with its length indicating number of entity storage elements of the discrete-event system. The  $N$ th element of the vector defines an entity storage element with index  $N$ . Use utility methods such as `queueFIFO` to create such definition as a MATLAB structure.

**I** — Connections between input ports and entity storage elements

vector

Define connections between input ports and entity storage elements as a vector. The length of the vector must match the number of input ports of this discrete-event system. The *N*th element of the vector defines the connection between the *N*th input port and any entity storage element. If the input port is an entity port, a valid entity storage index must be specified. If the input port is a signal port, the element takes a value of zero.

Do not connect multiple entity input ports to a common storage element. As a result, all nonzero elements of this vector must have distinct values.

## Examples

### Specify Entity Storage Elements

Specify entity storage elements and connections between entity input ports and storage elements for the discrete-event system object.

```
function [storageSpecs, I] = getEntityStorageImpl(obj)
    % Specify entity storage elements and connections between
    % entity input ports and storage elements.
    %
    % The implementation specifies two storage elements for the
    % discrete-event system:
    % 1. A priority queue
    %   - Stores entities of type 'student'
    %   - Has maximal capacity of 25
    %   - Sort entities by an attribute named 'age', in ascending
    %     direction
    % 2. A FIFO queue
    %   - Stores entities of type 'student'
    %   - Has maximal capacity of 10
    %   - Sort entities in a First-In-First-Out order
    %
    % The implementation also specifies that the entity input port
    % of the discrete-event system is connected to the 2nd storage
    % element.
    %
    % Other methods of the discrete-event system must have defined:
    % - An entity type named 'student' (by method 'getEntityTypesImpl')
    % - An entity input port (by method 'getEntityPortsImpl')
    %
    storageSpecs = [...
        obj.queuePriority('student', 25, 'age', 'ascending'), ...
```



```
        obj.queueFIFO('student', 10)];  
I = 2;  
end
```

- [Simevents Examples](#)

## See Also

[matlab.DiscreteEventSystem](#) | [matlab.DiscreteEventSystem.getEntityTypesImpl](#) |  
[matlab.DiscreteEventSystem.queueFIFO](#) | [matlab.DiscreteEventSystem.queueLIFO](#)  
| [matlab.DiscreteEventSystem.queuePriority](#) |  
[matlab.DiscreteEventSystem.queueSysPriority](#)

## More About

- [“Discrete-Event Systems Created with a System Object”](#)

**Introduced in R2016a**

## getEntityTypesImpl

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Define entity types of discrete-event system

### Syntax

```
entityTypes=getEntityTypesImpl(obj)
```

### Description

`entityTypes=getEntityTypesImpl(obj)` defines entity types of a discrete-event system.

### Input Arguments

**obj** — Discrete-event System object

MATLAB object

Discrete-event System object.

### Output Arguments

**entityTypes** — Entity types

vector of MATLAB structures

Entity types returned as a vector whose length is the same as the number of entity types. Each vector element is a structure containing the entity type properties:

- Name
- Data dimension
- Data type

- Complexity

## Examples

### Get Entity Types

Get entity types *entity1* and *entity2* for discrete-event system, *obj*.

```
function entityTypes = getEntityTypesImpl(obj)
    % Define entity type 'type1' with inherited data type, dimension
    % and complexity
    t1 = obj.entityType('type1');

    % Define entity type 'type2' with specified data type ('mybus'),
    % default dimension and complexity (i.e. scalar real values)
    t2 = obj.entityType('type2', 'mybus');

    % Define entity type 'type3' with specified data type ('double'),
    % dimension (2 by 3 matrix), and complexity (complex)
    t3 = obj.entityType('type3', 'double', [2 3], true);

    entityTypes = [t1, t2, t3];
end
```

- Simevents Examples

### See Also

matlab.DiscreteEventSystem | matlab.DiscreteEventSystem.entityType  
| matlab.DiscreteEventSystem.getEntityPortsImpl |  
matlab.DiscreteEventSystem.getEntityStorageImpl

### More About

- “Discrete-Event Systems Created with a System Object”

Introduced in R2016a

## iterateImpl

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Event action when entity iterates

## Syntax

```
[entity,events,next]=iterateImpl(obj,storage,entity,tag,cur)
[entity,events,next,out1,...]=iterateImpl(obj,storage,entity,tag,
cur,in1,...)
```

## Description

[entity,events,next]=iterateImpl(obj,storage,entity,tag,cur) specifies event actions for when an entity is processed as a part of an iterate event.

[entity,events,next,out1,...]=iterateImpl(obj,storage,entity,tag,cur,in1,...) specifies such event actions when the block has one or more input signal ports and/or signal output ports.

## Input Arguments

**obj** — Discrete-event System object

MATLAB object

Discrete-event System object.

**storage** — Storage

double

Index of the storage element.

**entity** — Entity

MATLAB structure

Entity currently being processed.

**tag — Tag**

string

Tag of the current entity iterate event.

**cur — Current state**

MATLAB structure

MATLAB structure indicating current state of iteration. The structure has these fields:

- **size**  
Total number of entities the storage has
- **position**  
Position of the current iterating entity

**in1 — Signal input**

any value

Any data inputs of the object. These input arguments exist only when the object has data inputs.

## Output Arguments

**entity — Entity**

MATLAB structure

Entity being processed, possibly with changed data.

**events — Events**

vector of MATLAB structures

Events to be scheduled after the method returns.

**next — Iteration**

logical | double

- True

Continue to process the next entity in the storage element.

- False

Terminate the iterate event, and leave the rest of the entities of the storage element unprocessed.

### **out1 – Signal output**

any value

Data outputs of the object. You must specify these output arguments when the object has data outputs.

## **Examples**

### **Forward the First Entity**

Forward the first entity with matching data value to output port 1 of the discrete-event system.

```
function [entity,events,next] = iterateImpl(obj,storage,entity,tag,status)
% Forward the first entity with matching data value to output
% port 1 of the discrete-event system.
disp(['Searching in storage element ' num2str(storage)]);
disp(['    Total size = ' num2str(status.size)]);
disp(['    Current position = ' num2str(status.position)]);
if (entity.data == obj.dataToSearch)
    events = obj.eventForward('output', 1, 0);
    next = false;    % Found -- early terminate
else
    events = [];
    next = true;    % Not yet found -- continue
end
end
```

- Simevents Examples

### **See Also**

matlab.DiscreteEventSystem | matlab.DiscreteEventSystem.blockedImpl |  
matlab.DiscreteEventSystem.destroyImpl | matlab.DiscreteEventSystem.entryImpl  
| matlab.DiscreteEventSystem.exitImpl | matlab.DiscreteEventSystem.generateImpl

| matlab.DiscreteEventSystem.getEntityPortsImpl |  
matlab.DiscreteEventSystem.getEntityStorageImpl  
| matlab.DiscreteEventSystem.getEntityTypesImpl |  
| matlab.DiscreteEventSystem.setupEventsImpl |  
matlab.DiscreteEventSystem.timerImpl

## **More About**

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**

## queueFIFO

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Define first-in first-out (FIFO) queue storage

### Syntax

```
storage=queueFIFO(entityType,capacity)
```

### Description

`storage=queueFIFO(entityType,capacity)` defines a FIFO queue storage element. Use this function when implementing the `matlab.DiscreteEventSystem.getEntityStorageImpl` method.

### Input Arguments

**entityType** — Entity type

string

Type of entities that the new storage element works with.

**capacity** — Maximum number of entities

double

Maximum number of entities that the storage can contain, specified as a double.

### Output Arguments

**storage** — Storage

MATLAB structure

Queue storage that contains entities and sorts them in FIFO order.



## Examples

### Specify FIFO Queue Entity Storage

Specify FIFO queue entity storage for the discrete-event system object.

```
% Define a storage element as a FIFO queue
% - Entities in the queue are sorted in First-In-First-Out (FIFO) order
% - Queue can store entities of type 'myEntity'
% - Queue can store no more than 25 entities
storage = obj.queueFIFO('myEntity', 25);
```

- Simevents Examples

### See Also

matlab.DiscreteEventSystem.getEntityStorageImpl |  
matlab.DiscreteEventSystem.queueLIFO | matlab.DiscreteEventSystem.queuePriority |  
matlab.DiscreteEventSystem.queueSysPriority

### More About

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**

## queueLIFO

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Define last-in last-out (LIFO) queue storage

### Syntax

```
storage=queueLIFO(entityType,capacity)
```

### Description

`storage=queueLIFO(entityType,capacity)` defines a LIFO queue storage element. Use this function when implementing the `matlab.DiscreteEventSystem.getEntityStorageImpl` method.

### Input Arguments

**entityType** — Entity type

string

Type of entities that the new storage element works with.

**capacity** — Capacity

double

Maximum number of entities that the storage can contain, specified as a double.

### Output Arguments

**storage** — Storage

MATLAB structure

Queue storage that contains entities and sorts them in a LIFO order.

## Examples

### Define LIFO Queue Storage

Define LIFO queue storage.

```
% Define a storage element as a LIFO queue
% - Entities in the queue are sorted in Last-In-First-Out (LIFO) order
% - Queue can store entities of type 'myEntity'
% - Queue can store no more than 25 entities
storage = obj.queueLIFO('myEntity', 25);
```

- Simevents Examples

### See Also

matlab.DiscreteEventSystem.getEntityStorageImpl |  
matlab.DiscreteEventSystem.queueFIFO | matlab.DiscreteEventSystem.queuePriority |  
matlab.DiscreteEventSystem.queueSysPriority

### More About

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**

## queuePriority

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Define priority queue storage

### Syntax

```
storage=queueLIFO(entityType,capacity,key,order)
```

### Description

`storage=queueLIFO(entityType,capacity,key,order)` defines a priority queue that sorts entities by custom attribute. Use this function when implementing the `matlab.DiscreteEventSystem.getEntityStorageImpl` method.

### Input Arguments

**entityType** — Entity type

string

Type of entities that the new storage element works with.

**capacity** — Capacity

double

Maximum number of entities that the storage can contain, specified as a double.

**key** — Key

string

Name of the attribute that is used as the key for sorting.

**order** — Sorting order

string

Direction of sorting. Specify 'ascending' if you want entities with smaller key values to appear in front of the queue. Specify 'descending' if you want entities with greater key values to appear in front of the queue.

## Output Arguments

**storage** — Storage  
MATLAB structure

Queue storage element that contains entities and sorts them using a custom attribute.

## Examples

### Define Storage Element as a Priority Queue

Define storage element as a priority queue.

```
% Define a storage element as a priority queue
% - Queue sorts entities using a specific attribute of the entities
% - Queue can store entities of type 'myEntity'
% - Queue can store no more than 25 entities
% - Queue uses the attribute 'age' to sort entities
% - Sorting direction is 'ascending', resulting entities with
%   smaller 'age' attribute values to appear in front of the queue
storage = obj.queuePriority('myEntity', 25, 'age', 'ascending');
```

- Simevents Examples

### See Also

matlab.DiscreteEventSystem.getEntityStorageImpl |  
matlab.DiscreteEventSystem.queueFIFO | matlab.DiscreteEventSystem.queueLIFO |  
matlab.DiscreteEventSystem.queueSysPriority

### More About

- “Discrete-Event Systems Created with a System Object”

Introduced in R2016a

## queueSysPriority

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Define system priority queue storage

### Syntax

```
storage=queueSysPriority(entityType,capacity,order)
```

### Description

`storage=queueSysPriority(entityType,capacity,order)` defines a priority queue storage element that sorts entities by their system priorities. Use this method when implementing the `matlab.DiscreteEventSystem.getEntityStorageImpl` method.

### Input Arguments

**entityType** — Entity type

string

Type of entities that the new storage element works with.

**capacity** — Capacity

double

Maximum number of entities that the storage can contain, specified as a double.

**order** — Sorting order

string

Direction of sorting. Specify 'ascending' if you want entities with smaller system priority values (higher priority) to appear in front of the queue. Use 'descending' if you want entities with higher system priority values (lower priority) to appear in front of the queue.

## Output Arguments

### storage — Storage

MATLAB structure

Queue storage element that contains entities and sorts them by the entities' system priorities.

## Examples

### Define Storage Element as System Priority Queue

Define a storage element that uses an entity system priority for sorting.

```
% - Queue sorts entities using entity priority (i.e.
%   the field 'entVar.sys.priority' on a MATLAB variable 'entVar'
%   representing a SimEvents entity)
% - Queue can store entities of type 'myEntity'
% - Queue can store no more than 25 entities
% - Sorting direction is 'ascending', resulting entities with
%   higher priority (or smaller entity priority values) to appear
%   in the front of the queue
storage = obj.queueSysPriority('myEntity', 25, 'ascending');
```

- Simevents Examples

### See Also

matlab.DiscreteEventSystem.getEntityStorageImpl |  
 matlab.DiscreteEventSystem.queueFIFO | matlab.DiscreteEventSystem.queueLIFO |  
 matlab.DiscreteEventSystem.queuePriority

### More About

- “Discrete-Event Systems Created with a System Object”

Introduced in R2016a

## setupEventsImpl

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Initialize entity generation events

### Syntax

```
events=setupEventsImpl(obj)
```

### Description

`events=setupEventsImpl(obj)` sets up the first set of entity generation events at the start of simulation.

### Input Arguments

**obj** — Discrete-event System object

MATLAB object

Discrete-event System object.

### Output Arguments

**events** — Events

vector of MATLAB structures

A vector of events to create initial entities. The discrete-event system schedules these events at the start of simulation.

### Examples

#### Schedule Two Entity Generation Events

Schedules two entity generation events at the start of the simulation



```
function events = setupEventsImpl(obj)
    % Schedules two entity generation events at the start of the
    % simulation
    % - An event with tag 'Adam' to generate an entity in storage element 1.
    % - An event with tag 'Eve' to generate an entity in storage element 2.
    events = [...
        obj.eventGenerate(1, 'Adam', 0.5, 200), ...
        obj.eventGenerate(2, 'Eve', 0.8, 100)];
end
```

- Simevents Examples

## See Also

matlab.DiscreteEventSystem | matlab.DiscreteEventSystem.blockedImpl |  
matlab.DiscreteEventSystem.destroyImpl | matlab.DiscreteEventSystem.entryImpl  
| matlab.DiscreteEventSystem.exitImpl | matlab.DiscreteEventSystem.generateImpl  
| matlab.DiscreteEventSystem.getEntityPortsImpl |  
matlab.DiscreteEventSystem.getEntityStorageImpl |  
matlab.DiscreteEventSystem.getEntityTypesImpl |  
matlab.DiscreteEventSystem.iterateImpl | matlab.DiscreteEventSystem.timerImpl

## More About

- “Discrete-Event Systems Created with a System Object”

**Introduced in R2016a**

## timerImpl

**Class:** matlab.DiscreteEventSystem

**Package:** matlab

Event action when timer completes

## Syntax

[entity,events]=timerImpl(obj,storage,entity,tag)

[entity,events,out1,...]=timerImpl(obj,storage,entity,tag,in1,...)

## Description

[entity,events]=timerImpl(obj,storage,entity,tag) specifies event actions for when scheduled timer completes.

[entity,events,out1,...]=timerImpl(obj,storage,entity,tag,in1,...) specifies such event actions when the block has one or more input signal ports and/or signal output ports.

## Input Arguments

**obj** — Discrete-event System object

MATLAB object

Discrete-event System object.

**storage** — Storage

double

Index of the storage element.

**entity** — Entity

MATLAB structure

Entity for the timer event.

**tag — Tag**

string

Tag of the currently executing timer event.

**in1 — Signal input**

any value

Any data inputs of the object. These input arguments exist only when the object has data inputs.

## Output Arguments

**entity — Entity**

MATLAB structure

Entity with changed value.

**events — Events**

vector of MATLAB structures

Events to be scheduled after the method returns.

**out1 — Signal output**

any value

Data outputs of the object. You must specify these output arguments when the object has data outputs.

## Examples

**Event Action When Timer Completes**

Forward entity when timer completes for discrete-event system object *obj*.

```
function [entity,events] = timerImpl(obj,storage,entity,tag)
    % Check which timer of the entity has expired, and forward the
    % entity to the next location accordingly.
    switch tag
        case 'ServiceComplete'
```

```
        entity.done = 1;
        events = obj.eventForward('output', 1, 0);
    case 'Timeout'
        entity.done = 0;
        events = obj.eventForward('storage', 2, 0);
    end
end
```

- [Simevents Examples](#)

## See Also

[matlab.DiscreteEventSystem](#) | [matlab.DiscreteEventSystem.blockedImpl](#) | [matlab.DiscreteEventSystem.destroyImpl](#) | [matlab.DiscreteEventSystem.entryImpl](#) | [matlab.DiscreteEventSystem.exitImpl](#) | [matlab.DiscreteEventSystem.generateImpl](#) | [matlab.DiscreteEventSystem.getEntityPortsImpl](#) | [matlab.DiscreteEventSystem.getEntityStorageImpl](#) | [matlab.DiscreteEventSystem.getEntityTypesImpl](#) | [matlab.DiscreteEventSystem.iterateImpl](#) | [matlab.DiscreteEventSystem.setupEventsImpl](#)

## More About

- [“Discrete-Event Systems Created with a System Object”](#)

**Introduced in R2016a**

# simevents

Open SimEvents library

## Syntax

```
simevents  
simevents('1')  
simevents('2')  
simevents('3')  
simevents('4')  
simevents('5')
```

## Description

simevents opens the main SimEvents® library.

simevents('1') opens version 1.2 of the SimEvents library.

simevents('2') and simevents('3') open version 3.1.2 of the SimEvents library.

simevents('4') opens version 4.4.1 of the SimEvents library.

simevents('5') opens version 5.0 of the SimEvents library.

**Introduced in R2011b**

## **simeventslib**

Open SimEvents library

### **Syntax**

```
simeventslib
```

### **Description**

simeventslib opens the main SimEvents library.

**Introduced before R2006a**

# simevents.SimulationObserver class

**Package:** simevents

**Superclasses:** handle

Interface to create your custom visualization for models with SimEvents blocks

## Description

This class is an interface for creating custom visualizations for models with SimEvents blocks. Subclass this class to create your own visualization, using the methods below. Some utility functions are also provided to interact with event calendars, blocks, and entities. Do not overwrite these utility functions.

## Construction

`obj = SimulationObserver(modelName)` returns an object of the `SimulationObserverclass`, used to create a model observer for a SimEvents model.

## Input Arguments

**modelName** — Model to observe

string

The name of the model to observe.

## Methods

### Public Methods to Implement

<code>simevents.SimulationObserver.simStarted</code>	Specify behavior when simulation starts
<code>simevents.SimulationObserver.simPaused</code>	Specify behavior when simulation pauses
<code>simevents.SimulationObserver.simResumed</code>	Specify behavior when simulation resumes
<code>simevents.SimulationObserver.simTerminating</code>	Define observer behavior when simulation is terminating
<code>simevents.SimulationObserver.getBlocksToNotify</code>	Specify list of blocks to be notified of entity entry and exit events

<code>simevents.SimulationObserver.notifyEventCalendarWhen</code>	Specify whether you want notification for all events in event calendar
<code>simevents.SimulationObserver.postEntry</code>	Specify behavior after an entity enters a block that has entity storage
<code>simevents.SimulationObserver.preExit</code>	Specify behavior before an entity exits a block with entity storage
<code>simevents.SimulationObserver.preExecute</code>	Specify behavior before execution of an event

## Utility Functions

<code>simevents.SimulationObserver.addBlockNotification</code>	Add block to list of blocks to be notified
<code>simevents.SimulationObserver.removeBlockNotification</code>	Remove block from list of blocks being notified
<code>simevents.SimulationObserver.getEventCalendars</code>	Get handles to event calendars
<code>simevents.SimulationObserver.getAllBlockWithStorages</code>	Get list of blocks that store entities
<code>simevents.SimulationObserver.getHandleToBlock</code>	Block handle for a given block path
<code>simevents.SimulationObserver.getHandlesToBlockStorages</code>	Block storage handles of specified block

## Examples

### Construct Animator

Construct an animator.

```
function this = seExampleRestaurantAnimator
    % Constructor
    modelName = 'seExampleCustomVisualization';
    this@simevents.SimulationObserver(modelName);
    this.mModel = modelName;
end
```

- Simevents Examples

### See Also

`simevents.SimulationObserver.addBlockNotification` |  
`simevents.SimulationObserver.getAllBlockWithStorages`  
| `simevents.SimulationObserver.getBlocksToNotify` |  
`simevents.SimulationObserver.getEventCalendars` |  
`simevents.SimulationObserver.getHandleToBlock` |  
`simevents.SimulationObserver.getHandlesToBlockStorages` |



simevents.SimulationObserver.postEntry  
simevents.SimulationObserver.preExit  
simevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

## More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

## addBlockNotification

**Class:** simevents.SimulationObserver

**Package:** simevents

Add block to list of blocks to be notified

### Syntax

```
addBlockNotification(obj,blkPath)
```

### Description

`addBlockNotification(obj,blkPath)` is a utility function for adding a block to the list of blocks to be notified. Specify the full path of the block to be added in `blkPath`.

### Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

**blkPath** — Full path of the block to be notified

string

Full path of the block to be added to the list of blocks to be notified.

### Examples

#### Add Block to List of Blocks for Notification

Add block to list of blocks for notification.

```
function postEntry(obj,eventSource,eventData)
    if someConditionIsTrue
```

```

        addBlockNotification(obj,[this.mModel '/Patron Enter']);
    end
end

```

- Simevents Examples

## See Also

[simevents.SimulationObserver.addBlockNotification](#) |  
[simevents.SimulationObserver.getAllBlockWithStorages](#)  
[| simevents.SimulationObserver.getBlocksToNotify](#) |  
[simevents.SimulationObserver.getEventCalendars](#) |  
[simevents.SimulationObserver.getHandleToBlock](#) |  
[simevents.SimulationObserver.getHandlesToBlockStorages](#) |  
[simevents.SimulationObserver.postEntry](#)[simevents.SimulationObserver.preExit](#)[simevents.SimulationObserver.removeBlockNotification](#) |  
[simevents.SimulationObserver.simPaused](#) | [simevents.SimulationObserver.simResumed](#)  
[| simevents.SimulationObserver.simStarted](#) |  
[simevents.SimulationObserver.simTerminating](#)

## More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

## getAllBlockWithStorages

**Class:** simevents.SimulationObserver

**Package:** simevents

Get list of blocks that store entities

### Syntax

```
getAllBlockWithStorages(obj)
```

### Description

`getAllBlockWithStorages(obj)` is a utility function that returns the paths of all blocks that store entities.

### Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

### Output Arguments

**allBlockPaths** — Paths of all blocks that store entities

cell array of strings

Cell array of all blocks that store entities, provided as full block paths.

### Examples

**Return Paths of All Blocks that Store Entities**

Return the paths of all blocks that store entities.

```
function blks=getBlocksToNotify(obj)
    blks=getAllBlockWithStorages(obj);
end
```

- Simevents Examples

## See Also

simevents.SimulationObserver.addBlockNotification |  
simevents.SimulationObserver.getAllBlockWithStorages  
| simevents.SimulationObserver.getBlocksToNotify |  
simevents.SimulationObserver.getEventCalendars |  
simevents.SimulationObserver.getHandleToBlock |  
simevents.SimulationObserver.getHandlesToBlockStorages |  
simevents.SimulationObserver.postEntry  
simevents.SimulationObserver.preExit  
simevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

## More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

## getBlocksToNotify

**Class:** simevents.SimulationObserver

**Package:** simevents

Specify list of blocks to be notified of entity entry and exit events

### Syntax

```
getBlocksToNotify(obj)
```

### Description

`getBlocksToNotify(obj)` is used to specify a cell array of block paths that are notified by the `SimulationObserver` object. These blocks have to be discrete event blocks with entity storages. Override this function in your subclass to specify a cell array of blocks for which `simevents.SimulationObserver.preExit` and `simevents.SimulationObserver.postEntry` methods will be called. Specify the string 'ALL' to run these methods on all the discrete-event blocks with entity storages in the model. If you do not want any blocks to be notified, specify an empty cell array, `{}`.

### Input Arguments

**obj** — **SimulationObserver object**

string

Object of class `SimulationObserver`

### Output Arguments

**b1ks** — **List of blocks being notified of runtime events**

`{}` (default) | cell array of strings

Cell array of full block paths of all blocks being notified of runtime events

## Examples

### Blocks to Observe in Model

Return the list of blocks you want to observe in the model.

```
function blks = getBlocksToNotify(this)
    % Return list of blocks to observe in the model
    %
    % For this example, we are only interested in the following
    % blocks as they are sufficient for us to know all events of
    % interest
    blks = { ...
        [this.mModel '/Patron Enter'], ...
        [this.mModel '/Have Dinner'], ...
        [this.mModel '/Patron Leave'] ...
    };
end
```

- Simevents Examples

### See Also

simevents.SimulationObserver.addBlockNotification |  
simevents.SimulationObserver.getAllBlockWithStorages  
| simevents.SimulationObserver.getBlocksToNotify |  
simevents.SimulationObserver.getEventCalendars |  
simevents.SimulationObserver.getHandleToBlock |  
simevents.SimulationObserver.getHandlesToBlockStorages |  
simevents.SimulationObserver.postEntry  
simevents.SimulationObserver.preExit  
simevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

### More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

## getEventCalendars

**Class:** simevents.SimulationObserver

**Package:** simevents

Get handles to event calendars

### Syntax

```
getEventCalendars(obj)
```

### Description

`getEventCalendars(obj)` is a utility method that returns handles to all event calendars in your model.

### Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

### Output Arguments

**evCa1** — Array of event calendars in model

array of handles to EventCalendar objects

Array of handles to the event calendars in your model.

### Examples

#### Get Handles to Event Calendars in Model

Get handles to all event calendars in your model.



```
function postEntry(obj, evSrc, evData)
    % Print simulation time
    evcal=getEventCalendas(obj);
    tNow=evcal(1).timeNow;
    disp(tNow);
end
```

- Simevents Examples

## See Also

simevents.SimulationObserver.addBlockNotification |  
simevents.SimulationObserver.getAllBlockWithStorages  
| simevents.SimulationObserver.getBlocksToNotify |  
simevents.SimulationObserver.getEventCalendars |  
simevents.SimulationObserver.getHandleToBlock |  
simevents.SimulationObserver.getHandlesToBlockStorages |  
simevents.SimulationObserver.postEntry  
simevents.SimulationObserver.preExit  
simevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

## More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

# getHandlesToBlockStorages

**Class:** simevents.SimulationObserver

**Package:** simevents

Return storage handles of specified block

## Syntax

getHandlesToBlockStorages(obj, blkPath)

## Description

getHandlesToBlockStorages(obj, blkPath) returns the storage handles for the block specified by blkPath. If the block does not store entities, this method returns a 0x0 array of simevents.Storage objects.

## Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

**blkPath** — Full path to block

string

Full path to the block that stores entities

## Output Arguments

**storagesForBlock** — Storage handles for the block

array of handles to simevents.Storage objects

Array of storage handles of the block. If the block does not store entities, output is a 0x0 array of storage.

## Examples

### Get Handles for All Block Storage Elements

Get handles for all block storage elements in the model.

```
function postEntry(obj, evSrc, evData)
    % Number of entities in server;
    storage=getHandlesToBlockStorages(obj, [this.mModel ' /Have Dinner' ]);
    disp(length(storage.Entity));
end
```

- Simevents Examples

### See Also

simevents.SimulationObserver.addBlockNotification |  
simevents.SimulationObserver.getAllBlockWithStorages  
| simevents.SimulationObserver.getBlocksToNotify |  
simevents.SimulationObserver.getEventCalendars |  
simevents.SimulationObserver.getHandleToBlock |  
simevents.SimulationObserver.getHandlesToBlockStorages |  
simevents.SimulationObserver.postEntry  
simevents.SimulationObserver.preExit  
simevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

### More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

## getHandleToBlock

**Class:** simevents.SimulationObserver

**Package:** simevents

Return block handle for a given block path

### Syntax

```
getHandleToBlock(obj,blkPath)
```

### Description

`getHandleToBlock(obj,blkPath)` is a utility function that returns the handle to the block whose full path is specified by `blkPath`.

### Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

**blkPath** — Full path to block

string

### Output Arguments

**blkHandle** — Handle to block

handle to block

Handle to the block specified in `blkPath`.

## Examples

### Return Handle to Specified Block

Return handle to specified block.

```
function postEntry(obj, evSrc, evData)
   hdl=getHandleToBlock(obj, [this.mModel '/Have Dinner']);
    ...
end
```

- Simevents Examples

### See Also

simevents.SimulationObserver.addBlockNotification |  
simevents.SimulationObserver.getAllBlockWithStorages  
| simevents.SimulationObserver.getBlocksToNotify |  
simevents.SimulationObserver.getEventCalendars |  
simevents.SimulationObserver.getHandleToBlock |  
simevents.SimulationObserver.getHandlesToBlockStorages |  
simevents.SimulationObserver.postEntry  
simevents.SimulationObserver.preExit  
simevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

### More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

# notifyEventCalendarEvents

**Class:** simevents.SimulationObserver

**Package:** simevents

Specify whether you want notification for all events in event calendar

## Syntax

```
notifyEventCalendarEvents(obj)
```

## Description

`notifyEventCalendarEvents(obj)` specifies whether you want notification for all events in the event calendar before they are executed. Set the output of this method to `true` to call the `simevents.SimulationObserver.preExecute` method for all events in the event calendar.

## Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

## Output Arguments

**n** — Boolean specifying whether all events in event calendar are notified before executing

false (default) | true

Boolean that specifies whether you are notified of all events in the event calendar before executing. If set to `true`, the `simevents.SimulationObserver.preExecute` method is called for every event before its execution.

## Examples

### Specify Notification for All Events in Event Calendar

Specify whether you want notification for all events in event calendar.

```
function status=notifyEventCalendarEvents(obj)
    status=false;
end
```

- [Simevents Examples](#)

### See Also

[simevents.SimulationObserver.addBlockNotification](#) |  
[simevents.SimulationObserver.getAllBlockWithStorages](#)  
[| simevents.SimulationObserver.getBlocksToNotify](#) |  
[simevents.SimulationObserver.getEventCalendars](#) |  
[simevents.SimulationObserver.getHandleToBlock](#) |  
[simevents.SimulationObserver.getHandlesToBlockStorages](#) |  
[simevents.SimulationObserver.postEntry](#) | [simevents.SimulationObserver.preExit](#) | [simevents.SimulationObserver.removeBlockNotification](#) |  
[simevents.SimulationObserver.simPaused](#) | [simevents.SimulationObserver.simResumed](#)  
[| simevents.SimulationObserver.simStarted](#) |  
[simevents.SimulationObserver.simTerminating](#)

### More About

- [“Interface for Custom Visualization”](#)

**Introduced in R2016a**

## postEntry

**Class:** simevents.SimulationObserver

**Package:** simevents

Specify behavior after an entity enters a block that has entity storage

## Syntax

```
postEntry(obj, evSrc, evData)
```

## Description

`postEntry(obj, evSrc, evData)` is used to specify behavior after an entity enters a block that has entity storage. The simulation observer uses this method as a callback for post-entry event notification and provides handles to the entity, the block and its storage, and the event.

## Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

**evSrc** — Handle to block storage

handle to simevents.Storage object

Handle to block storage in which the entity entered. The handle will be populated by the simulation observer.

**evData** — List of handles for block, storage, and entities, and event type

cell array of handles

List of handles for block, storage, and entities. The list will be populated by the simulation observer.



## Examples

### Specify Listener for Storage Entry

Specify listener to execute when entity enters a storage element such as a queue or server.

```
function postEntry(this, evSrc, evData)
    % Override to specify listener for entry into a storage (queue/server)

    entity = evData.CurrentEntity;

    if strcmp(evData.Block.BlockPath, [this.mModel '/Have Dinner'])

        % Identify which table the customer is going to
        tblId = this.occupyTable(entity);

        % Schedule motion for this customer to the appropriate
        % table
        target = this.cTablePos(tblId, :);
        this.scheduleMotion(entity, target);

        % Decrement the waiting statistic
        this.updateStats(this.mTxtWaiting, this.DECREMENT);

    elseif strcmp(evData.Block.BlockPath, [this.mModel '/Patron Leave'])
        % Schedule motion for this entity from its current position
        % to the exit position
        if isKey(this.mEntityGlyphs, num2str(entity.ID))
            this.scheduleMotion(entity, this.cExitPos);
        end

        % Schedule for the entity dot to be destroyed when it has
        % completed its pending motion
        this.scheduleMotion(entity, [NaN, NaN]);

    end
end
```

- Simevents Examples

## See Also

simevents.SimulationObserver.addBlockNotification |  
simevents.SimulationObserver.getAllBlockWithStorages  
| simevents.SimulationObserver.getBlocksToNotify |  
simevents.SimulationObserver.getEventCalendars |  
simevents.SimulationObserver.getHandleToBlock |  
simevents.SimulationObserver.getHandlesToBlockStorages |  
simevents.SimulationObserver.postEntry  
simevents.SimulationObserver.preExit  
simevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

## More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

## preExecute

**Class:** simevents.SimulationObserver

**Package:** simevents

Specify behavior before execution of an event

## Syntax

```
preExecute(obj, evSrc, evData)
```

## Description

`preExecute(obj, evSrc, evData)` is used to specify behavior before the execution of an event in the event calendar. The simulation observer uses this method as a callback for pre-execute event notifications and provides a handle to the event calendar.

## Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

**evSrc** — Handle to event calendar

handle to simevents.EventCalendar object

Handle to event calendar. The handle will be populated by the simulation observer.

**evData** — Event name and handle to event calendar

cell array of handles

Event name and handle to event calendar

## Examples

### Specify Behavior Before Execution of Event

Specify behavior before the execution of an event in the event calendar.

```
function preExecute(obj, evSrc, evData)
    fprintf('Specify behavior before the execution of an event in the event calendar.')
end
```

- Simevents Examples

### See Also

simevents.SimulationObserver.addBlockNotification |  
simevents.SimulationObserver.getAllBlockWithStorages  
| simevents.SimulationObserver.getBlocksToNotify |  
simevents.SimulationObserver.getEventCalendars |  
simevents.SimulationObserver.getHandleToBlock |  
simevents.SimulationObserver.getHandlesToBlockStorages |  
simevents.SimulationObserver.postEntry  
simevents.SimulationObserver.preExit  
simevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

### More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

# preExit

**Class:** simevents.SimulationObserver

**Package:** simevents

Specify behavior before an entity exits a block with entity storage

## Syntax

```
preExit(obj, evSrc, evData)
```

## Description

`preExit(obj, evSrc, evData)` is used to specify behavior before an entity exits a block that stores entities. The simulation observer uses this method as a callback for pre-exit event notification and provides handles to the entity, the block and its storage, and the event.

## Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

**evSrc** — Handle to event calendar

handle to simevents.EventCalendar object

Handle to event calendar. The handle will be populated by the simulation observer.

**evData** — List of handles of block, storage, and entities, and event type

cell array of handles

List of handles of block, storage, and entities. The list will be populated by the simulation observer.

## Examples

### Specify Listener for Storage Exit

Specify listener to execute when entity exits a storage element such as a queue or server.

```
function preExit(this, ~, evData)
    % Override to specify listener for exit from a storage (queue/server)
    % evData contains block, storage, and entity handles

    entity = evData.CurrentEntity;

    if strcmp(evData.Block.BlockPath, [this.mModel '/Patron Enter'])
        % Create a new "dot" on the figure at the entry position
        h = plot(this.cEntryPos(1), this.cEntryPos(2), '.');
        set(h, 'MarkerSize', 32);

        % Add a mouse-click function to the dot so we can retrieve
        % attribute data when user clicks on this customer
        set(h, 'ButtonDownFcn', @(h,e)entityClickFcn(this,h,e));

        % Cache away the entity identifier on this dot
        set(h, 'Tag', num2str(entity.ID));

        % Cache away this dot handle so that we can move it in
        % future events
        this.mEntityGlyphs(num2str(entity.ID)) = h;

        % Cache away the entity handle
        this.mEntities(num2str(entity.ID)) = entity;

        % Increment the entry statistics
        this.updateStats(this.mTxtEntry, this.INCREMENT);

        % Schedule motion for this entity from its current position
        % to a random position in the waiting area
        this.scheduleMotion(entity, this.getRandWaitingPos());

        % Increment waiting statistic
        this.updateStats(this.mTxtWaiting, this.INCREMENT);
    elseif strcmp(evData.Block.BlockPath, [this.mModel '/Have Dinner'])
        this.releaseTable(entity);
    end
end
```

end

- Simevents Examples

## See Also

simevents.SimulationObserver.addBlockNotification |  
simevents.SimulationObserver.getAllBlockWithStorages  
| simevents.SimulationObserver.getBlocksToNotify |  
simevents.SimulationObserver.getEventCalendars |  
simevents.SimulationObserver.getHandleToBlock |  
simevents.SimulationObserver.getHandlesToBlockStorages |  
simevents.SimulationObserver.postEntry  
simevents.SimulationObserver.preExit  
simevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

## More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

## removeBlockNotification

**Class:** simevents.SimulationObserver

**Package:** simevents

Remove block from list of blocks being notified

### Syntax

```
removeBlockNotification(obj,blkPath)
```

### Description

`removeBlockNotification(obj,blkPath)` is a utility function used to remove a block from the list of blocks being notified. Specify the full path of the block to be added in `blkPath`.

### Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

**blkPath** — Full path of the block to be notified

string

Full path of the block to be added to the list of blocks being notified.

### Examples

#### Remove Block

Remove block from list of blocks being notified.

```
function postEntry(obj,eventSource,eventData)
```



```
if someConditionIsTrue
    removeBlockNotification(obj,[this.mModel '/Patron Enter']);
end
end
```

- [Simevents Examples](#)

## See Also

[simevents.SimulationObserver.addBlockNotification](#) |  
[simevents.SimulationObserver.getAllBlockWithStorages](#)  
| [simevents.SimulationObserver.getBlocksToNotify](#) |  
[simevents.SimulationObserver.getEventCalendars](#) |  
[simevents.SimulationObserver.getHandleToBlock](#) |  
[simevents.SimulationObserver.getHandlesToBlockStorages](#) |  
[simevents.SimulationObserver.postEntry](#)[simevents.SimulationObserver.preExit](#)[simevents.SimulationObserver.simPaused](#) |  
[simevents.SimulationObserver.removeBlockNotification](#) |  
[simevents.SimulationObserver.simPaused](#) | [simevents.SimulationObserver.simResumed](#)  
| [simevents.SimulationObserver.simStarted](#) |  
[simevents.SimulationObserver.simTerminating](#)

## More About

- [“Interface for Custom Visualization”](#)

**Introduced in R2016a**

## simPaused

**Class:** simevents.SimulationObserver

**Package:** simevents

Specify behavior when simulation pauses

## Syntax

`simPaused(obj)`

## Description

`simPaused(obj)` determines the behavior when the simulation is paused. Override this function to specify the behavior of your visualization when the simulation pauses, as determined by the `SimulationStatus` parameter.

## Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

## Examples

### Call Method When Pausing Model

Call this method when model is paused.

```
function simPaused(this)
    % Called when model is paused

    % Schedule the timer to stop when all pending animation is
    % completed
    this.mTimerRequestPause = true;
```

end

- Simevents Examples

## See Also

simevents.SimulationObserver.addBlockNotification |  
simevents.SimulationObserver.getAllBlockWithStorages  
| simevents.SimulationObserver.getBlockstoNotify |  
simevents.SimulationObserver.getEventCalendars |  
simevents.SimulationObserver.getHandleToBlock |  
simevents.SimulationObserver.getHandlesToBlockStorages |  
simevents.SimulationObserver.postEntry  
simevents.SimulationObserver.preExit  
simevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

## More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

## simResumed

**Class:** simevents.SimulationObserver

**Package:** simevents

Specify behavior when simulation resumes

## Syntax

`simResumed(obj)`

## Description

`simResumed(obj)` determines the behavior when the simulation resumes after pausing. Override this function to specify the behavior of your visualization when the simulation resumes, as determined by the `SimulationStatus` parameter.

## Input Arguments

**obj** — **SimulationObserver** object

string

Object of class `SimulationObserver`

## Examples

### Call Method When Model Continues

Call this method when model continues after pausing.

```
function simResumed(this)
    % Called when model continues from being paused

    % Restart the timer
    this.mTimerRequestPause = false;
    start(this.mTimer);
```

end

- Simevents Examples

## See Also

simevents.SimulationObserver.addBlockNotification |  
simevents.SimulationObserver.getAllBlockWithStorages  
| simevents.SimulationObserver.getBlocksToNotify |  
simevents.SimulationObserver.getEventCalendars |  
simevents.SimulationObserver.getHandleToBlock |  
simevents.SimulationObserver.getHandlesToBlockStorages |  
simevents.SimulationObserver.postEntry  
simevents.SimulationObserver.preExit  
simevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

## More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

## simStarted

**Class:** simevents.SimulationObserver

**Package:** simevents

Specify behavior when simulation starts

## Syntax

```
simStarted(obj)
```

## Description

`simStarted(obj)` determines the behavior when the simulation starts. Override this function to specify the behavior of your visualization when the simulation starts, as determined by the `SimulationStatus` parameter.

## Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

## Examples

### Initialize Animation Canvas

Initialize the animation canvas.

```
function simStarted(this)
    % Initialize the animation canvas

    % Re-initialize runtime work variables for simulation
    this.mEntityGlyphs = containers.Map('keytype', 'char', 'valuetype', 'any');
    this.mEntities = containers.Map('keytype', 'char', 'valuetype', 'any');
    this.mCombineMap = containers.Map('keytype', 'char', 'valuetype', 'char');
    this.mCachePostRun = containers.Map('keytype', 'char', 'valuetype', 'char');
    this.mTableOccupy = zeros(1, size(this.cTablePos,1)) - 1;
```

```

% Setup the figure with the restaurant floor as background
close all;
im = imread('restaurant.png');
image(im);
this.mFig = gcf;
set(this.mFig, 'Tag', 'Begin');
this.mAx = gca;
set(this.mFig, 'toolbar', 'none');
set(this.mFig, 'menubar', 'none');
set(this.mAx, 'XTickLabel', '');
set(this.mAx, 'YTickLabel', '');
set(this.mAx, 'Box', 'on');
set(this.mAx, 'TickLength', [0 0]);
set(this.mAx, 'position', [0 0 1 1]);
hold on;

% Set up the numeric statistics text labels on the figure
this.mTxtEntry = text(170,850, '0');
this.mTxtWaiting = text(10,160, '0');
this.mTxtExit = text(920,330, '0');
this.mTxtSelectedEnt = text(50,600, '');

set(this.mTxtEntry, 'Color', [0.8500 0.3250 0.0980], 'FontWeight', 'bold', 'FontSize', 14);
set(this.mTxtWaiting, 'Color', [0.8500 0.3250 0.0980], 'FontWeight', 'bold', 'FontSize', 14);
set(this.mTxtExit, 'Color', [0.8500 0.3250 0.0980], 'FontWeight', 'bold', 'FontSize', 14);

this.mLineSelectedEnt = plot(0,0, '.');

% Set up the timer
this.mTimer = timer(...
    'TimerFcn', @(t,e)animate(this,t,e), ...
    'ExecutionMode', 'fixedSpacing', ...
    'Period', this.cTimerPeriod);
this.mTimerData = containers.Map('keytype', 'char', 'valuetype', 'any');
this.mTimerRequestStop = false;
this.mTimerRequestPause = false;
start(this.mTimer);
end

```

- Simevents Examples

## See Also

[simevents.SimulationObserver.addBlockNotification](#) |  
[simevents.SimulationObserver.getAllBlockWithStorages](#)  
[| simevents.SimulationObserver.getBlocksToNotify](#) |  
[simevents.SimulationObserver.getEventCalendars](#) |  
[simevents.SimulationObserver.getHandleToBlock](#) |  
[simevents.SimulationObserver.getHandlesToBlockStorages](#) |  
[simevents.SimulationObserver.postEntry](#) [simevents.SimulationObserver.preExits](#) [simevents.SimulationObserver.removeBlockNotification](#) |  
[simevents.SimulationObserver.removeBlockNotification](#) |  
[simevents.SimulationObserver.simPaused](#) | [simevents.SimulationObserver.simResumed](#)  
[| simevents.SimulationObserver.simStarted](#) |  
[simevents.SimulationObserver.simTerminating](#)

## **More About**

- “Interface for Custom Visualization”

**Introduced in R2016a**



# simTerminating

**Class:** simevents.SimulationObserver

**Package:** simevents

Define observer behavior when simulation is terminating

## Syntax

```
simTerminating(obj)
```

## Description

`simTerminating(obj)` determines the behavior when the simulation is terminating. Override this function to specify the behavior of your visualization when the simulation is terminating, as determined by the `SimulationStatus` parameter.

## Input Arguments

**obj** — SimulationObserver object

string

Object of class SimulationObserver

## Examples

### Call Method When Simulation is Terminating

Call this method when simulation is terminating.

```
function simTerminating(this)
    % Called when simulation is terminating
    %
    % After the simulation terminates, in order to support clicking
    % on entity to see attributes, we gather up all of the entities
    % that exist in the model and save their attribute information
```

```
ents = this.mEntityGlyphs.keys;
for idx = 1 : length(ents)
    ent = ents{idx};
    try
        enStruct = this.mEntities(ent);
        str = evalc('disp(enStruct.Attributes)');
        this.mCachePostRun(ent) = str;
    catch me
    end
end

% If animation timer is still running, schedule a stop
if strcmp(this.mTimer.Running, 'on')
    this.mTimerRequestStop = true;
else
    % If timer is not running, delete it
    delete(this.mTimer);
end
end
```

- Simevents Examples

## See Also

simevents.SimulationObserver.addBlockNotification |  
simevents.SimulationObserver.getAllBlockWithStorages  
| simevents.SimulationObserver.getBlocksToNotify |  
simevents.SimulationObserver.getEventCalendars |  
simevents.SimulationObserver.getHandleToBlock |  
simevents.SimulationObserver.getHandlesToBlockStorages |  
simevents.SimulationObserver.postEntrysimevents.SimulationObserver.preExitsimevents.SimulationObserver.removeBlockNotification |  
simevents.SimulationObserver.simPaused | simevents.SimulationObserver.simResumed  
| simevents.SimulationObserver.simStarted |  
simevents.SimulationObserver.simTerminating

## More About

- “Interface for Custom Visualization”

**Introduced in R2016a**

# Blocks — Alphabetical List

---

## Attribute Function (Obsolete)

Access and modify attributes using MATLAB code

### Library

Attributes



This block accepts an entity, assigns data to it, and then outputs it. Assigned data is stored in attributes of the entity, where each attribute has a name and a value.

This block corresponds to a function that you write in an editor window that opens when you double-click the block. Your function names the attributes you want to access, modify, or create. When writing your function, you can use any part of the MATLAB language that is suitable for code generation, subject to the argument-naming rules described in “Write Functions to Manipulate Attributes” and the attribute support described in “Attribute Value Support”.

---

**Note:** If you attach large arrays to entities in a model that contains a server or a queue block with large capacity, the simulation could run out of memory.

---

### Timing and Connections

In most cases, it is not necessary to introduce a storage block between the **Attribute Function** block and subsequent blocks that use attributes (for example, **Attribute Scope**). However, the next table indicates exceptional cases in which you should insert a **Single Server** block between the **Attribute Function** block and the block performing the subsequent operation.

Subsequent Operation	Block
Switching based on the same attribute that the Attribute Function block created or modified	Output Switch block with <b>Switching criterion=From attribute</b>
Preemption based on the same attribute that the Attribute Function block created or modified	Single Server block with <b>Permit preemption based on attribute</b> selected

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities.

### Entity Output Ports

Label	Description
OUT	Port for entities whose attributes the block accessed, created, or modified.

## Examples

- “Set Attributes”
- “Incorporate Legacy Code”

## See Also

Set Attribute (Obsolete), Get Attribute (Obsolete)

“Write Functions to Manipulate Attributes”

**Introduced in R2007b**

## Attribute Scope (Obsolete)

Plot data from attribute of arriving entities

### Library



### Description

This block creates a plot using data from a real scalar-valued attribute of arriving entities. Use the **Y attribute name** parameter to specify which attribute to plot along the vertical axis.

Use the **Enable entity OUT port** option to choose whether the entity advances to a subsequent block or whether the block absorbs the arriving entity.

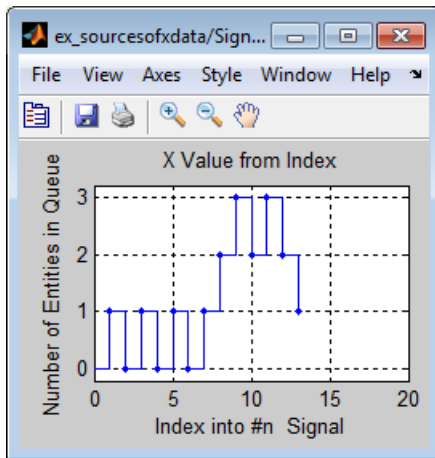
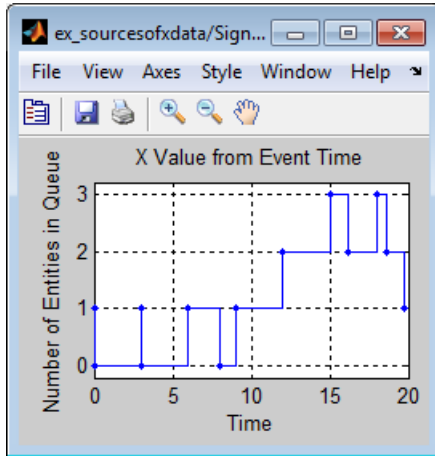
The **Plot type** parameter on the **Plotting** determines whether and how the block connects the points that it plots.

### Selecting Data for the Horizontal Axis

Use the **X value from** parameter to select the type of data for the horizontal axis. The table below describes the choices.

Source of X Data	Description of Plot
Event time	Plot of the specified attribute versus simulation time.
Index	Plot of the successive values of the specified attribute against a horizontal axis that represents the index of the values. The first entity's attribute value has an index of 1, the second entity's attribute value has an index of 2, and so on. For example, you might use this option when multiple entities might arrive simultaneously, to help determine the exact sequence among the simultaneous attribute values.

The figures below illustrate the different sources of data for the horizontal axis. The plots look similar, except that the second plot has uniform horizontal spacing rather than time-based spacing between successive points.



## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities, whose attributes contain the data to plot.

### Entity Output Ports

Label	Description
OUT	Port for departing entities. You see this port only if you select <b>Enable entity OUT port</b> .

### Signal Output Ports

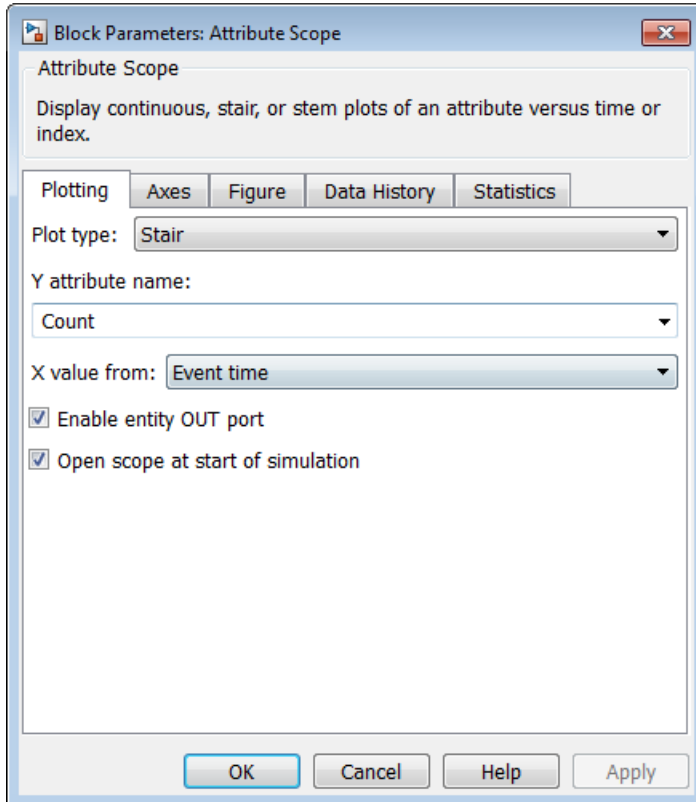
Label	Description
#a	Number of entities that have arrived at the block since the start of the simulation.

## Dialog Box

To open the block dialog box, click the Parameters toolbar button in the plot window.



## Plotting



### Plot type

The presentation format for the data.

### Y attribute name

Name of the attribute to plot along the vertical axis.

### X value from

Source of data for the plot's horizontal axis. See “Selecting Data for the Horizontal Axis” on page 2-4 for details.

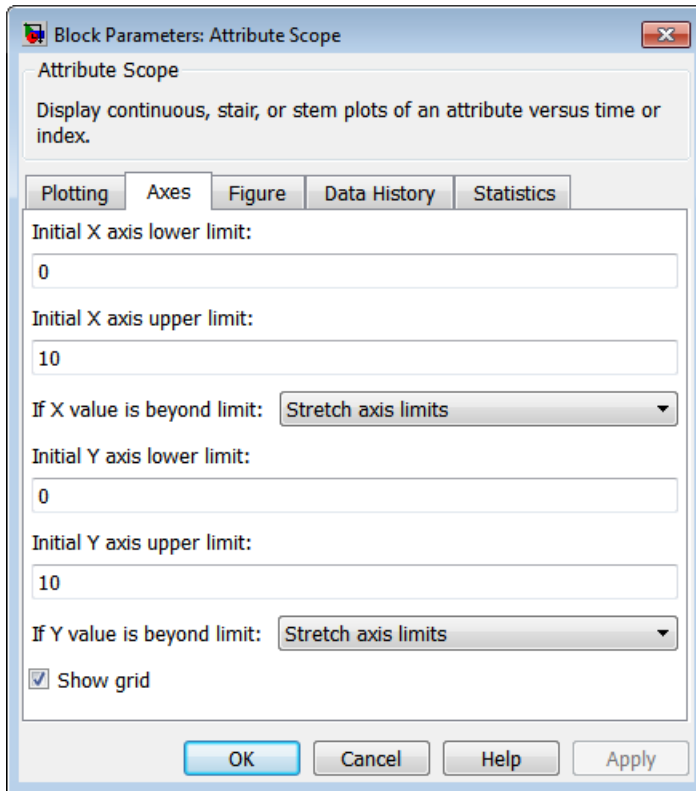
### Enable entity OUT port

Causes the block to have an entity output port labeled **OUT**, through which the arriving entity departs. If you clear this box, the block absorbs arriving entities.

### Open scope at start of simulation

Selecting this option causes the plot window to open when you start the simulation. If you clear this box, you can open the plot window by double-clicking the block icon.

### Axes



### Initial X axis lower limit, Initial X axis upper limit

The interval shown on the X axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If X value is beyond limit** setting.

### If X value is beyond limit

Determines how the plot changes if one or more X values are not within the limits shown on the X axis.

### Initial Y axis lower limit, Initial Y axis upper limit

The interval shown on the Y axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If Y value is beyond limit** setting.

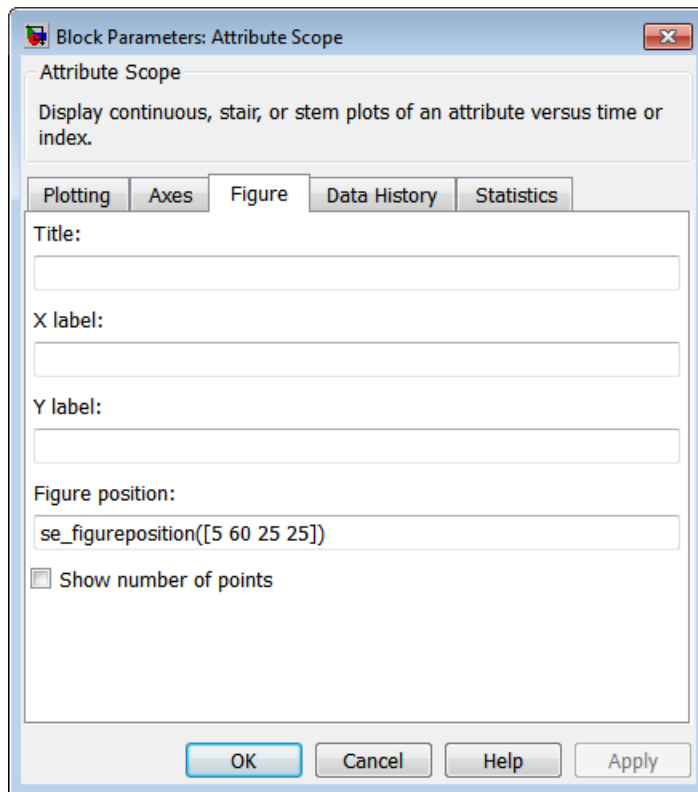
### If Y value is beyond limit

Determines how the plot changes if one or more attribute values are not within the limits shown on the Y axis.

### Show grid

Toggles the grid on and off.

## Figure



### Title

Text that appears as the title of the plot, above the axes.

**Y label**

Text that appears to the left of the vertical axis.

**X label**

Text that appears below the horizontal axis.

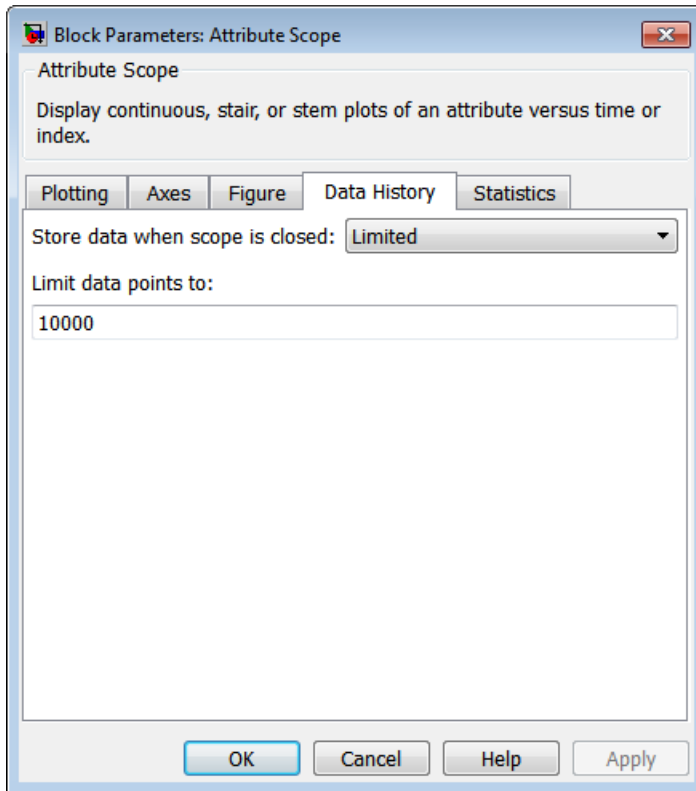
**Position**

A four-element vector of the form [left bottom width height] specifying the position of the scope window. (0,0) is the lower left corner of the display.

**Show number of entities**

Displays the number of plotted points using an annotation in the plot window.

**Data History**



### Store data when scope is closed

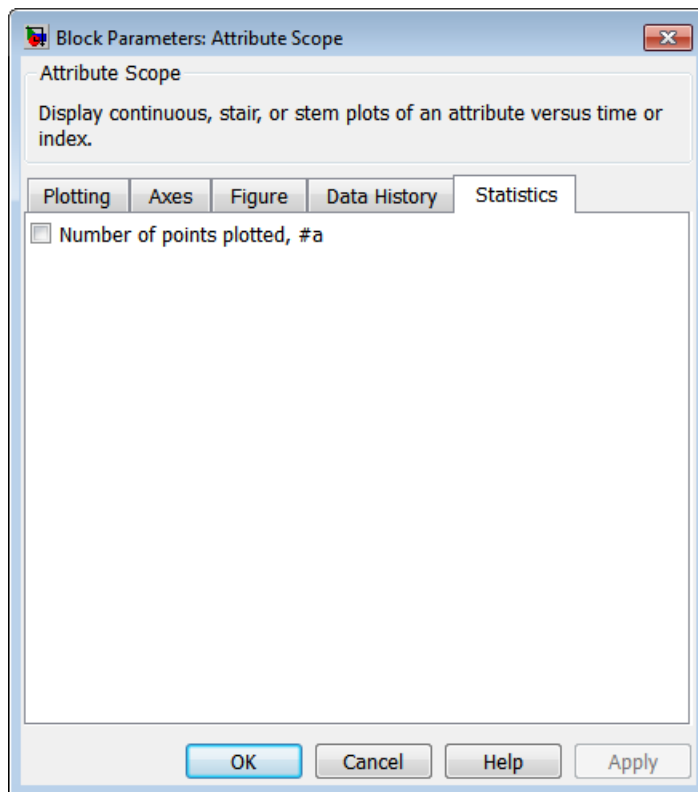
Select **Unlimited** to have the block cache all data for future viewing, **Limited** to cache a portion of the most recent data, and **Disabled** to avoid caching undisplayed data.

### Limit data points to

The number of data points the block caches, using the most recent data. **Store data when scope is closed** to **Limited**.

## Statistics

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see the preceding table, “Signal Output Ports”.



**Number of entities arrived**

**#a.**

## **Examples**

- “Set Attributes”

## **See Also**

X-Y Attribute Scope (Obsolete), Signal Scope (Obsolete)

“Manipulate Entity Attributes”

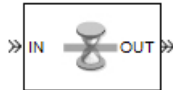
**Introduced before R2006a**

# Cancel Timeout (Obsolete)

Cancel timeout event for each entity

## Library

Timing



## Description

This block cancels a named timeout event that the **Schedule Timeout (Obsolete)** block previously scheduled for the arriving entity. Timeout events enable you to limit the time that an entity spends on designated entity paths during the simulation. Topologically, this block designates an end of an entity path that is relevant to the time limit. The ability to cancel timeout events before they occur lets you apply the time limit to an entity path that does not end with a sink block.

The **Timeout tag** parameter of this block is the name of the timeout event and corresponds to the **Timeout tag** parameter of a **Schedule Timeout** block in the model. If the arriving entity is not associated with a timeout event of that name, then you can configure the block to produce an error or warning, or to ignore the absence of the timeout event.

Using the **Residual time** and **Average residual time** parameters, you can configure the block to report the following statistics via the **rt** and **w** signal output ports, respectively:

- The residual time for the named timeout event associated with the arriving entity, which is the amount of time between the entity's arrival time at this block and the scheduled time of the named timeout event
- The average among the **rt** values among all entities that have arrived at this block during the simulation and been associated with timeouts of the specified name

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities.

### Entity Output Ports

Label	Description
OUT	Port for entities whose timeout event the block has just canceled.

### Signal Output Ports

Label	Description	Time of Update When Statistic is On	Order of Update
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	3
#t	Number of entities that have departed from this block and been associated with a timeout of the specified name.	After entity departure	2
rt	Amount of time between arrival time at this block and the scheduled time of the named timeout event.	After entity departure	2
w	Average among the <b>rt</b> values among all entities that have arrived at this block and been associated with timeouts of the specified name.	After entity departure	1

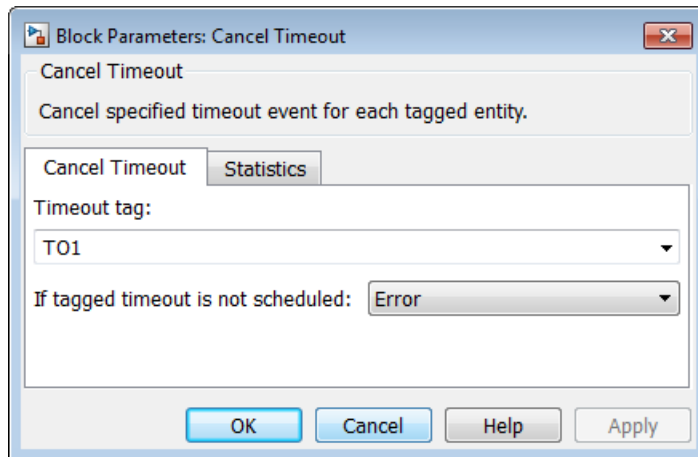
Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.



## Dialog Box

### Cancel Timeout Tab



#### Timeout tag

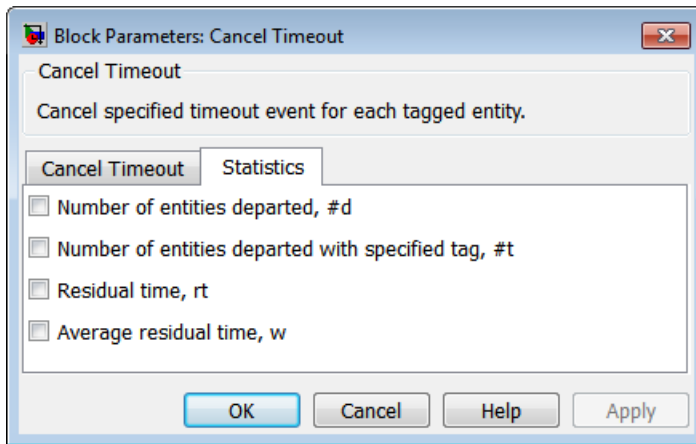
Name of the timeout event to cancel, corresponding to the **Timeout tag** parameter of a `Schedule Timeout` block in the model.

#### If tagged timeout is not scheduled

Behavior of the block if an arriving entity is not associated with a timeout event with the specified timeout tag.

### Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports.



### **Number of entities departed**

Allows you to use the signal output port labeled **#d**.

### **Number of entities departed with specified tag**

Allows you to use the signal output port labeled **#t**.

### **Residual time**

Allows you to use the signal output port labeled **rt**.

### **Average residual time**

Allows you to use the signal output port labeled **w**.

## **See Also**

Schedule Timeout (Obsolete)

**Introduced in R2007a**

## Conn (Obsolete)

Provide entity input port or entity output port for virtual subsystem

### Library

SimEvents Ports and Subsystems

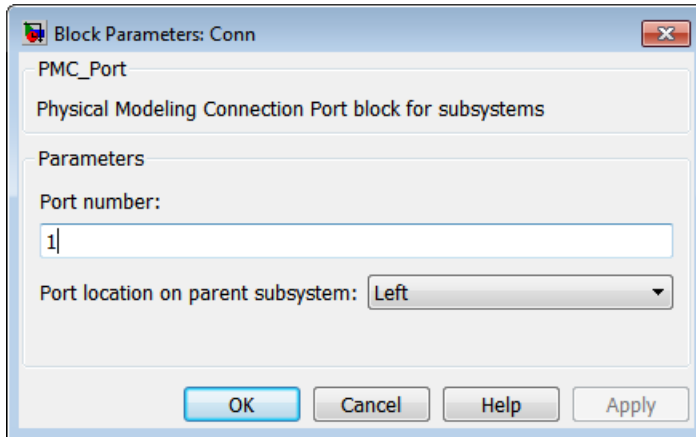
### Description

The Conn block, placed inside a subsystem containing blocks with entity ports, creates an entity port on the boundary of the subsystem. When you connect the Conn block, the port changes its appearance and becomes either an entity input port or an entity output port:

- Conn represents an input port if connected to another block's input port.
- Conn represents an output port if connected to another block's output port.

To create a new virtual subsystem, select one or more blocks in the model and select **Diagram > Subsystem & Model Reference > Create Subsystem from Selection** from the model window's menu. The application automatically inserts and connects appropriate input and output ports. To add more ports to the subsystem along entity paths, insert and connect this block in the subsystem window.

## Dialog Box



### Port number

Labels the subsystem connector port created by this block. Each connector port on the boundary of a single subsystem requires a unique number as a label.

### Port location on parent subsystem

Use this parameter to choose on which side of the parent subsystem boundary the port appears. The choices are **Left** and **Right**. The choice of port location is unrelated to whether the block represents an entity input port or an entity output port.

### Introduced in R2007b

# Discrete Event Signal to Workspace (Obsolete)

Write event-based signal to workspace

## Library

SimEvents Sinks



This block writes its input to a structure or array in the base MATLAB workspace when the simulation stops or pauses. One way to pause a running simulation is to select **Simulation > Pause**. Suspending the simulation during a debugger session does not cause this block to write data to the workspace. This block logs data at each sample time hit.

This block is similar to the `To Workspace` block in the Simulink<sup>®</sup> Sinks library but is tailored for use with event-based signals.

## Output Format

The **Save format** parameter determines the output format. The **Structure With Time** output format is most appropriate for event-based signals because it indicates when the signal assumes each value. Updates of event-based signals are typically aperiodic.

If the signal has an initial value, the value is the first data value in the workspace variable. (An example of a signal that has no initial value is a signal inside an Atomic Subsystem that has an event-based input signal.) Depending on the topology of your model, initial values of the signal might account for multiple data values in the workspace variable.

To identify the first data value that represents a sample time hit of the event-based signal, insert the **Initial Value** block before this block. Set the **Value until the first sample time hit** parameter to a value you do not expect as a signal value at a sample

time hit. To identify the initial value or values of the signal, remove the **Initial Value** block.

For scalar signals, you can convert a structure with time into a two-column matrix containing times in the first column and signal values in the second column. To do this, use an assignment like the one below. In place of `simout`, use the name specified in this block's **Variable name** parameter.

```
times_values = [simout.time, simout.signals.values];
```

For descriptions of all output formats, see the reference page for the **To Workspace** block.

### Comparison with To Workspace Block

This block has no **Sample time** parameter because event-based signals do not have a true sample time.

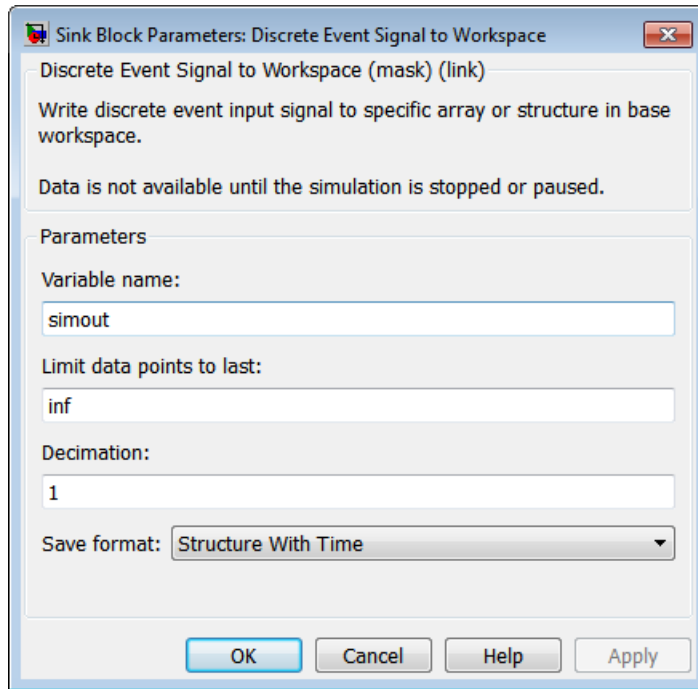
The simulation times at which this block records data is typically unrelated to the variable that a model creates if you select **Time** in the **Save to workspace** section of the **Data Import/Export** tab of the Configuration Parameters dialog box. By default, this option is selected and the variable is called `tout`.

### Ports

This block has one signal input port for the signal to write to the workspace.

The block has no entity ports, and no signal output port.

## Dialog Box



### Variable name

The name of the structure or array that holds the data.

### Limit data points to last

The maximum number of input samples to be saved.

### Decimation

A positive integer,  $n$ , that specifies the decimation factor. The block ignores the last  $n-1$  out of every  $n$  input samples.

### Save format

Format in which to save simulation output to the workspace. The recommended format for event-based signals is **Structure With Time**.

## **Examples**

- “Send Queue Length to the Workspace”

## **See Also**

To Workspace

“Save Simulation Data”

**Introduced before R2006a**



# Enabled Gate (Obsolete)

Allow entity arrivals upon positive control signal

## Library

Gates

## Description



This block represents a gate that is open whenever the control signal at the **en** input port is positive, and closed whenever the signal is zero or negative. By definition, an open gate permits entity arrivals as long as the entities would be able to advance immediately to the next block, while a closed gate forbids entity arrivals. The **en** signal is a numerical signal of type `double`. Because the signal can remain positive for a time interval of arbitrary length, an enabled gate can remain open for a time interval of arbitrary length. The length can be zero or a positive number.

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities.

### Signal Input Ports

Label	Description
en	The gate is open whenever this signal is positive. This signal must be an event-based signal.

### Entity Output Ports

Label	Description
OUT	Port for departing entities.

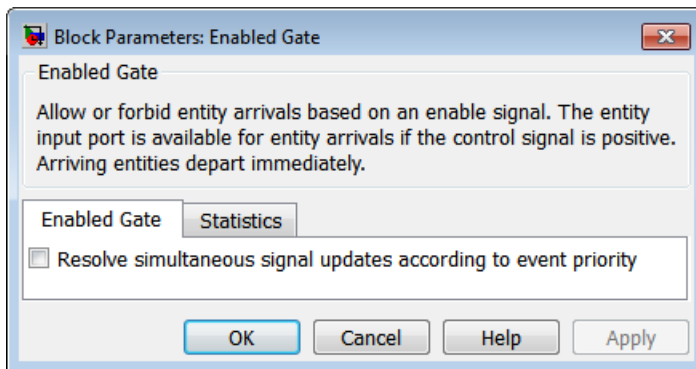
## Signal Output Ports

Label	Description	Time of Update When Statistic Is On
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure

The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

## Dialog Box

### Enabled Gate Tab



### Resolve simultaneous signal updates according to event priority

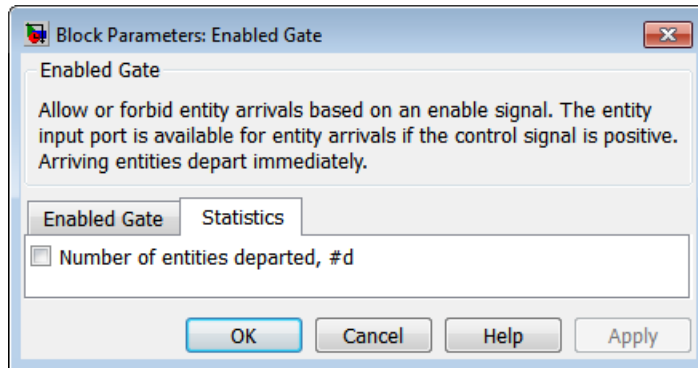
Select this option to prioritize the gate-opening or gate-closing event explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority SYS1 on the event calendar.

### Event priority

The priority of the gate-opening and gate-closing events, relative to other simultaneous events in the simulation. Gate opening and closing are distinct events that share the same event priority. You see this field only if you select **Resolve simultaneous signal updates according to event priority**.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see the preceding table, “Signal Output Ports”.



### Number of entities departed

Allows you to use the signal output port labeled **#d**.

## See Also

Release Gate (Obsolete)

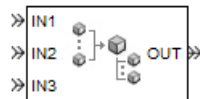
Introduced before R2006a

## Entity Combiner (Obsolete)

Generate one entity per set of entities arriving simultaneously

### Library

Entity Management



### Description

This block generates one new entity for each set of entities arriving simultaneously at multiple input ports. The arriving entities are called component entities. They might represent different parts within a larger item, such as a header, payload, and trailer that are parts of a packet. The **Entity Combiner** block and its preceding blocks automatically detect when all necessary component entities are ready for the combining operation to proceed. Your parameter choices in this block determine whether other blocks can access the attributes or timers of the component entities, and whether the combining operation is reversible. Some parameter choices require uniqueness of attribute names or timer tags in the component entities.

Timeout events, if any, corresponding to the component entities are canceled during the combining operation.

### Waiting for Component Entities on Multiple Paths

The **Entity Combiner** block has multiple entity input ports and one entity output port. The combining operation occurs when all necessary component entities are ready and the resulting entity would be able to depart. More explicitly, when all the blocks that connect to the **Entity Combiner** block's entity input ports have a pending entity simultaneously and the port connecting to the **Entity Combiner** block's entity output port is available, the **Entity Combiner** block accepts one entity arrival at each input port and outputs one entity. At all other times, the **Entity Combiner** block's input ports are unavailable.

---

**Tip** It is typical to connect a queue or other storage block to each entity input port of the **Entity Combiner** block. The storage blocks provide a place for pending entities to wait for other entity paths to have pending entities. Storage blocks are especially important if multiple component entities come from a single multiple-output block, such as a **Replicate** or **Entity Splitter** block.

---

## Managing Information When Combining Entities

The entity that departs from the **Entity Combiner** block can optionally carry information about the component entities that the block combines. In some applications, you might consider the information to be more important than the entities that carry it. The table below indicates how different options of the block produce different requirements and behavior regarding

- Uniqueness of attribute names among the entities at all entity input ports of the **Entity Combiner** block
- Uniqueness of timer tags among the entities at all entity input ports of the **Entity Combiner** block
- Your ability to use the departing entity to access attributes and timers from the component entities
- Your ability to split the departing entity into its components using the **Entity Splitter** block

---

**Note:** You can manage access to the set of attributes and the set of timers independently. The table treats attributes and timers together merely for conciseness.

---

### Options for Managing Information When Combining Entities

<ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Retain structure in departing entity</li> <li><input checked="" type="checkbox"/> Make attributes accessible in departing entity</li> <li><input checked="" type="checkbox"/> Make timers accessible in departing entity</li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> Retain structure in departing entity</li> <li><input checked="" type="checkbox"/> Copy attributes to departing entity</li> <li><input checked="" type="checkbox"/> Copy timers to departing entity</li> </ul>
<ul style="list-style-type: none"> <li>• You can split the departing entity, which is called a composite entity.</li> </ul>	<ul style="list-style-type: none"> <li>• You cannot split the departing entity.</li> <li>• Attribute names and timer tags must be unique.</li> </ul>

<ul style="list-style-type: none"> <li>• Attribute names and timer tags must be unique.</li> <li>• You can access attributes and timers.</li> </ul>	<ul style="list-style-type: none"> <li>• You can access attributes and timers.</li> </ul>
<input checked="" type="checkbox"/> Retain structure in departing entity <input type="checkbox"/> Make attributes accessible in departing entity <input type="checkbox"/> Make timers accessible in departing entity <ul style="list-style-type: none"> <li>• You can split the departing entity, which is called a composite entity. When you split the composite entity, attributes and timers of components become accessible.</li> <li>• Uniqueness of attribute names and timer tags is optional.</li> <li>• You cannot access attributes and timers via the departing entity.</li> </ul>	<input type="checkbox"/> Retain structure in departing entity <input type="checkbox"/> Copy attributes to departing entity <input type="checkbox"/> Copy timers to departing entity <ul style="list-style-type: none"> <li>• You cannot split the departing entity.</li> <li>• Uniqueness of attribute names and timer tags is optional.</li> <li>• You cannot access attributes and timers.</li> </ul>

If you do not select **Retain structure in departing entity**, you can think of the block as generating a new nonhierarchical entity, copying attribute or timer information to the new entity if necessary, and then discarding the component entities.

## Ports

### Entity Input Ports

Label	Description
IN1, IN2, IN3, and so on	Port for arriving entities. The <b>Number of entity input ports</b> parameter determines how many of these entity input ports the block has.

### Entity Output Ports

Label	Description
OUT	Port for departing entities.

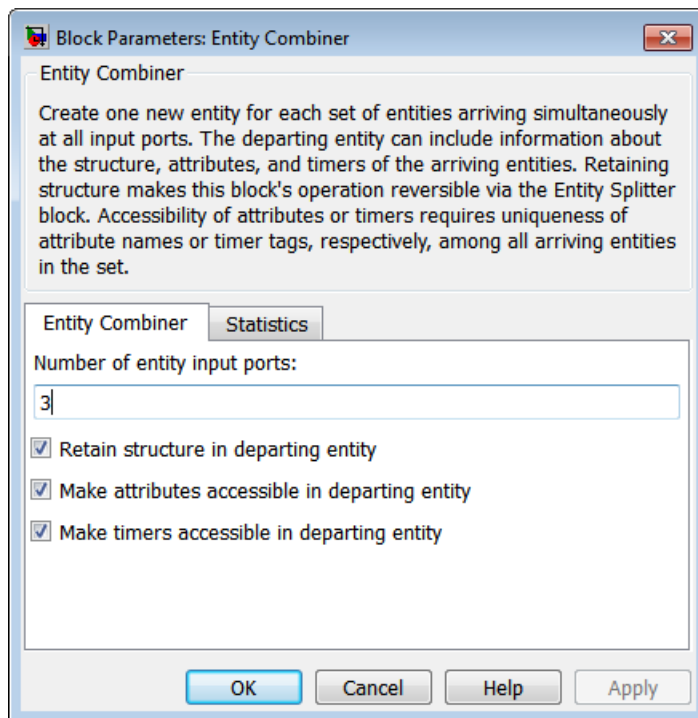
### Signal Output Ports

Label	Description	Time of Update When Statistic Is On
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure

The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

## Dialog Box

### Entity Combiner Tab



#### Number of entity input ports

Determines how many entity input ports the block has.

### Retain structure in departing entity

If you select this option, the departing entity carries information about the number of component entities and which attributes and timers each component entity possesses. Such information enables you to recover the component entities using the Entity Splitter block.

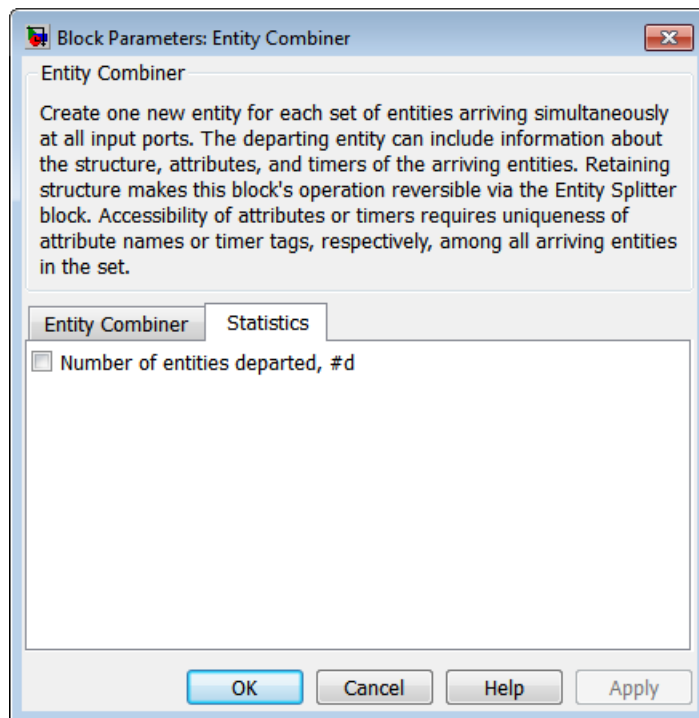
### Make attributes accessible in departing entity

If you select this option, you can access attributes from the component entities via the departing entity. The name of this field depends on whether you select **Retain structure in departing entity**.

### Make timers accessible in departing entity

If you select this option, you can access timers from the component entities via the departing entity. The name of this field depends on whether you select **Retain structure in departing entity**.

## Statistics Tab





### Number of entities departed

Allows you to use the signal output port labeled **#d**.

## Examples

- “Manage Data in Composite Entities”

## Limitations

In general, a composite entity can arrive at this block and become a component entity within a new nested composite entity. However, if you select **Retain structure in departing entity**, the depth of nesting is limited. This prevents the memory usage of nested composite entities from growing without bound in the case of a looped entity path.

## See Also

Entity Splitter (Obsolete)

“Combine Entities”

**Introduced in R2007a**

## Entity Departure Counter (Obsolete)

Count departures and write result to signal port or attribute

### Library

Entity Management



This block computes the number of entities that have departed since the start of the simulation or since the last reset, whichever occurred later. The block writes this number to a signal output port and/or an attribute of each departing entity. The count includes the departing entity.

### Ports

#### Entity Input Ports

Label	Description
IN	Port for arriving entities.

#### Signal Input Ports

Label	Description
ts	When this signal has an update, the block resets its internal counter and the <b>#d</b> output signal to zero. This signal must be an event-based signal. You see this port only if you set <b>Reset counter upon</b> to <b>Sample time hit from port ts</b> .
tr	When this signal satisfies the specified trigger criteria, the block resets its internal counter and the <b>#d</b> output signal to zero. This signal must be an event-based signal. You see this port only if you set <b>Reset counter upon</b> to <b>Trigger from port tr</b> .

Label	Description
<b>vc</b>	When this signal satisfies the specified value-change criteria, the block resets its internal counter and the <b>#d</b> output signal to zero. This signal must be an event-based signal. You see this port only if you set <b>Reset counter upon</b> to <b>Change in signal from port vc</b> .
<b>fcn</b>	When this signal carries a function call, the block resets its internal counter and the <b>#d</b> output signal to zero. This signal must be an event-based function call. You see this port only if you set <b>Reset counter upon</b> to <b>Function call from port fcn</b> .

### Entity Output Ports

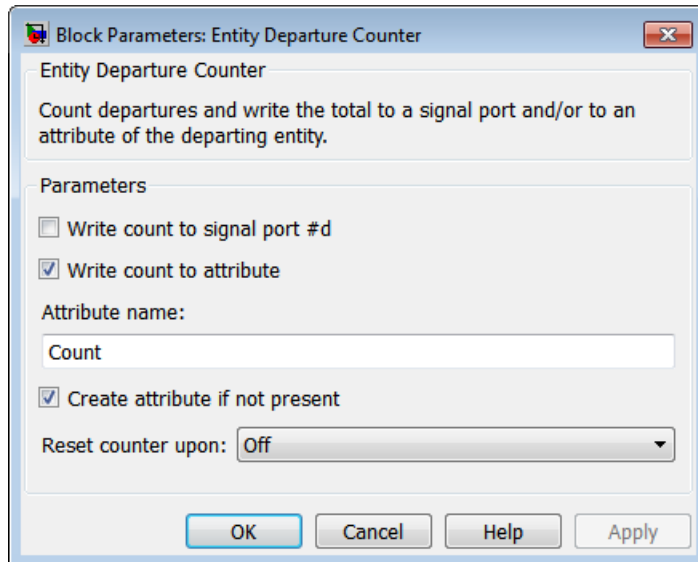
Label	Description
<b>OUT</b>	Port for departing entities.

### Signal Output Ports

Label	Description	Time of Update When Statistic Is On
<b>#d</b>	Number of entities that have departed from this block since the start of the simulation or since the last reset.	After entity departure

The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

## Dialog Box



### Write count to signal port #d

Allows you to use the signal output port labeled **#d**. This parameter determines whether the block outputs the entity count through a signal output port under these circumstances:

- Throughout the simulation
- Only when you stop or pause the simulation
- Not at all

### Write count to attribute

If you select this check box, the block assigns the entity count to the attribute specified in the **Attribute name** parameter.

### Attribute name

The name of the attribute the block uses to record the entity count. You see this field only if you select the **Write count to attribute** check box.

### Create attribute if not present

Selecting this option enables the block to define a new attribute for the entity count. Otherwise, the block issues an error if the attribute you name in the **Attribute name** parameter does not exist. You see this field only if select the **Write count to attribute** check box..

### Reset counter upon

Determines whether, and under which circumstances, the block resets its internal counter and the **#d** output signal to zero:

**Trigger type** determines whether rising, falling, or either type of trigger edge causes the counter to reset. You see this field only if you set **Reset counter upon** to Trigger from port tr.

**Type of change in signal value** determines whether rising, falling, or either type of value change causes the counter to reset. You see this field only if you set **Reset counter upon** to Change in signal from port vc.

### Resolve simultaneous signal updates according to event priority

Select this option to prioritize the reset event explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority SYS1 on the event calendar. You see this field only if you set **Reset counter upon** to a value other than Off.

### Event priority

The priority of the reset event, relative to other simultaneous events in the simulation. You see this field only if you set these parameters:

- **Reset counter upon** = A value other than Off
- Select **Resolve simultaneous signal updates according to event priority**

## Examples

- “Set Attributes”

## See Also

Instantaneous Entity Counting Scope (Obsolete)

“Count Entities”

**Introduced before R2006a**

# Entity Departure Function-Call Generator (Obsolete)

Convert entity departure event into one or two function calls

## Library

Generators/Function-Call Generators



This block converts an entity departure event into one or two function calls that you can use to invoke function-call subsystems, Stateflow<sup>®</sup> blocks, or other blocks that accept function-call inputs. The block can suppress its output under certain conditions.

## Criteria for Generating Function Calls

The primary criterion is the departure, or imminent departure, of an entity from the block. You can choose whether the block generates the function call before or after the departure.

To generate up to two function calls per event, select **Generate optional function call f2 after function call f1**. If you configure the block to generate the **f1** function call before the entity departure, you can independently choose whether the block generates the **f2** function call before or after that departure.

To make the **f1** or **f2** output function call contingent upon a secondary criterion, select **Suppress function call f1 if enable signal e1 is not positive** or **Suppress function call f2 if enable signal e2 is not positive**. The block acquires an additional signal input port, labeled **e1** or **e2**, to which you connect a control signal. If the control signal is zero or negative when the block is about to generate the function call, the block suppresses the function call. The **e1** and **e2** ports operate independently of each other as secondary criteria for their respective function-call output ports.

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities.

### Signal Input Ports

Label	Description
e1	When this signal is 0 or negative, the block does not generate a function call at the <b>f1</b> output port. This signal must be an event-based signal. You see this input port only if you select <b>Suppress function call f1 if enable signal e1 is not positive.</b>
e2	When this signal is 0 or negative, the block does not generate a function call at the <b>f2</b> output port. This signal must be an event-based signal. You see this input port only if you select <b>Suppress function call f2 if enable signal e2 is not positive.</b>

### Entity Output Ports

Label	Description
OUT	Port for departing entities.



## Signal Output Ports

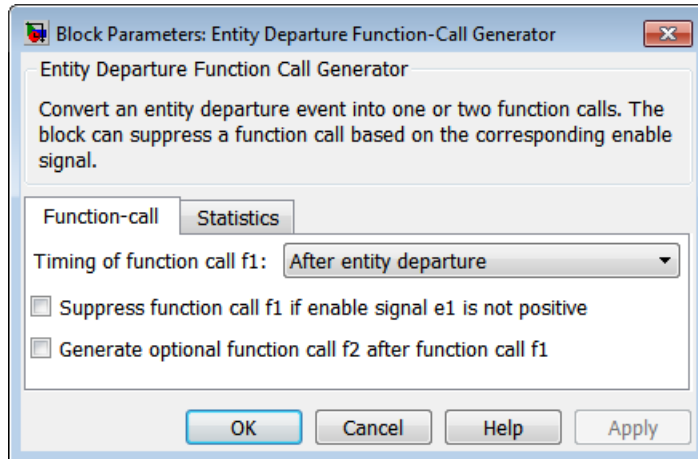
Label	Description	Time of Update When Port Is Present	Order of Update
<b>f1</b>	Function-call signal, possibly contingent on <b>e1</b> input signal.	Before or after entity departure, depending on <b>Timing of function call f1</b> parameter	1
<b>f2</b>	Function-call signal, possibly contingent on <b>e2</b> input signal.	Before entity departure if both <b>Timing of function call...</b> parameters are set to <b>Before entity departure</b> ; otherwise, after entity departure	2
<b>#d</b>	Number of entities that have departed from this block since the start of the simulation.	After entity departure	4
<b>#f1</b>	Number of function calls the block has generated at the <b>f1</b> port during the simulation.	After entity departure	3
<b>#f2</b>	Number of function calls the block has generated at the <b>f2</b> port during the simulation.	After entity departure	3

Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

The initial output value for the numerical signals, which is in effect from the start of the simulation until the first update by the block, is 0.

## Dialog Box

### Function Call Tab

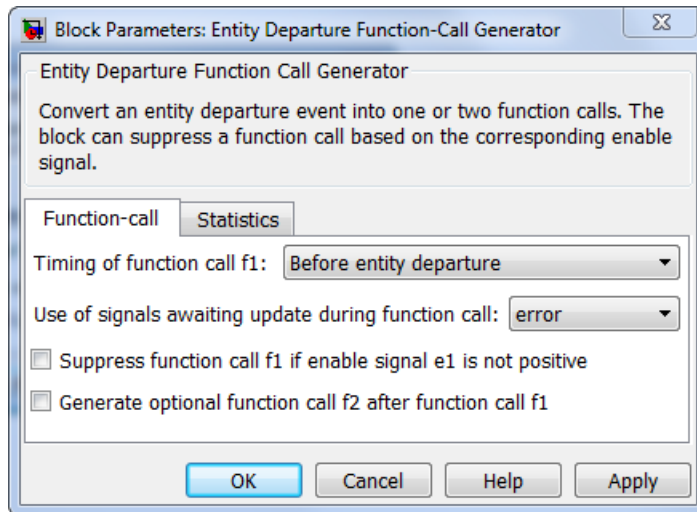


#### Timing of function call f1

Determines whether the **f1** function call occurs before or after the entity departure event.

#### Use of signals awaiting update during function call

You see this parameter only if you set **Timing of function call f1** to Before entity departure.



When you configure the block to generate a function-call before an entity departs the block, the function-call event might use signal values that are still awaiting update. If this situation arises, **Use of signals awaiting update during function call:** specifies the action that you want the software to take. If you do not also select the configuration parameter **Prevent duplicate events on multiport blocks and branched signals** in your model, the parameter does not function.

#### **Suppress function call f1 if enable signal e1 is not positive**

Selecting this option causes **f1** function calls to be contingent upon a positive value at the **e1** signal input port.

#### **Generate optional function call f2 after function call f1**

Selecting this option causes the block to generate a function call at the optional **f2** output port when appropriate criteria are satisfied.

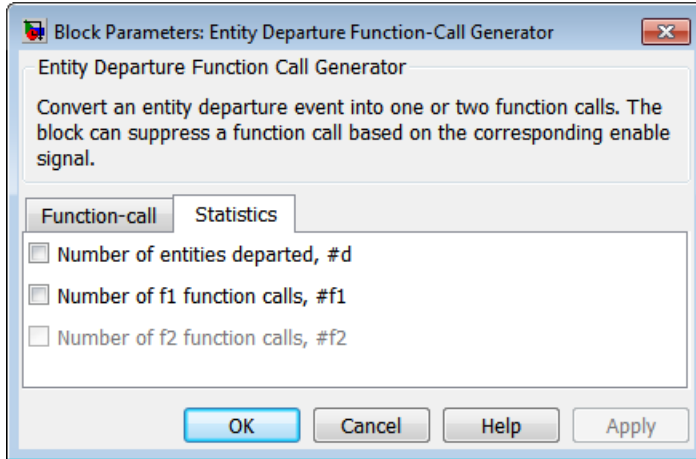
#### **Timing of function call f2**

Determines whether the **f2** function call occurs before or after the entity departure event. You see this field only if you set **Timing of function call f1** to **Before entity departure** and select **Generate optional function call f2 after function call f1**.

#### **Suppress function call f2 if enable signal e2 is not positive**

Selecting this option causes **f2** function calls to be contingent upon a positive value at the **e2** signal input port. You see this field only if you select **Generate optional function call f2 after function call f1**.

## Statistics Tab



### Number of entities departed

Allows you to use the signal output port labeled **#d**.

### Number of f1 function calls

Allows you to use the signal output port labeled **#f1**.

### Number of f2 function calls

Allows you to use the signal output port labeled **#f2**. This field is active only if you select **Generate optional function call f2 after function call f1** on the **Function Call** tab of this dialog box.

## See Also

Signal-Based Function-Call Generator (Obsolete)

Introduced in R2011b

# Composite Entity Creator

Create composite entities

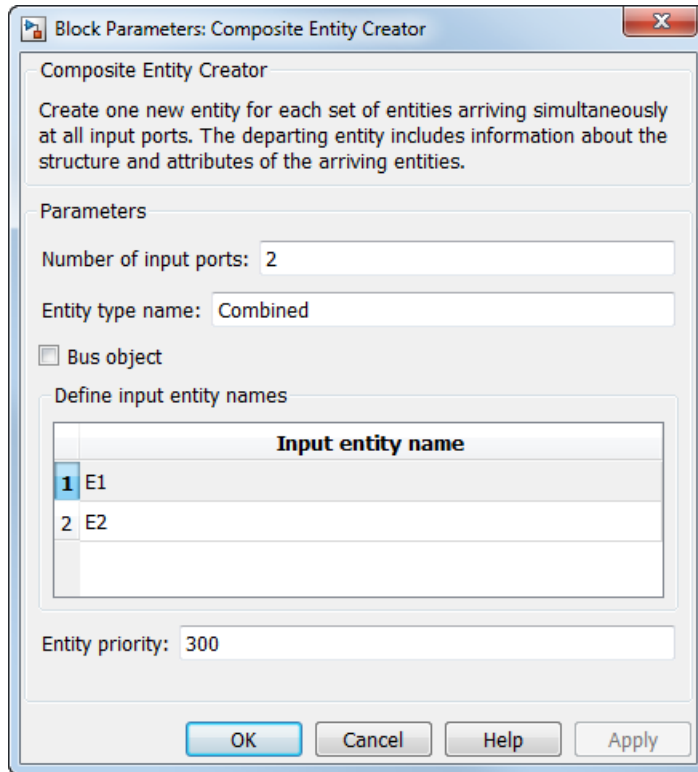
## Library

Entity Management



This block creates a composite entity for each set of entities arriving simultaneously at all input ports. The newly created entity can include information about the structure, attributes, and timers of the arriving entities.

## Dialog Box



### Number of input ports

Specify the number of input ports.

### Entity type name

Specify the type name of the composite entity that is created after combining incoming entities.

### Bus object

Specify whether to output the composite entity as a bus object.

### Define input entity names

Specify names to be attached to the input entities, which can be used for referencing these entities in the composite entity.

**Entity priority**

Specify the priority of the composite entity created.

**See Also**

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

**Related Examples**

- Simevents Examples

**Introduced in R2016a**

# Composite Entity Splitter

Split composite entities

## Library

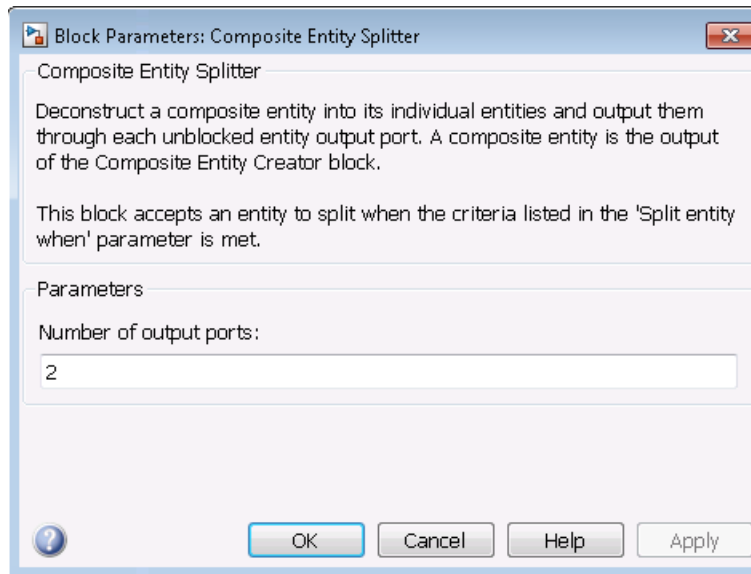
SimEvents Library: Routing



## Description

This block splits a composite entity into its individual entities and outputs them through each unblocked entity output port. A composite entity is the output of the Composite Entity Creator block.

## Dialog Box





### **Number of output ports**

Specify the number of output ports to output entities.

### **See Also**

Composite Entity Creator | Entity Generator

### **Related Examples**

- Simevents Examples

### **More About**

- “SimEvents Common Design Patterns”

**Introduced in R2016a**

# Discrete Event Chart

Discrete event chart

## Library

SimEvents Library: User-Defined

## Description

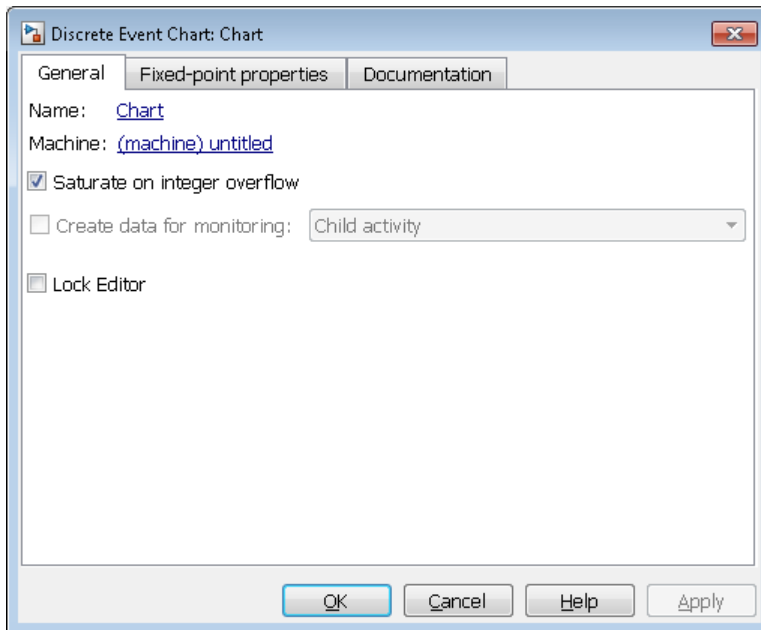


This block is similar to a Stateflow chart, but is used for discrete events. The advantages of the discrete event chart are given below.

- **Precise timing:** The time resolution for occurrence of events can be arbitrarily precise, and are not limited by the sample time of the model.
- **Trigger on Arrival:** A discrete event chart executes immediately on arrival of a message. It does not execute on the next sample time hit.
- **Dynamic Scheduling:** A discrete event chart can execute zero or multiple times in a single time step. It does not have a fixed sorted execution order. The order of execution depends on the run-time conditions of the model.

The Discrete Event Chart can be used in a similar fashion to the Stateflow Chart.

To access the properties of the chart, right-click the chart and select **Properties**. For more information on properties, see “Discrete Event Chart Properties”.



## See Also

Entity Generator | MATLAB Discrete-Event System

## Related Examples

- Simevents Examples

## More About

- “Discrete-Event Systems Created with Stateflow Charts”
- “Discrete Event Chart Properties”
- “SimEvents Common Design Patterns”

Introduced in R2016a

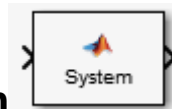
# MATLAB Discrete-Event System

MATLAB discrete-event system

## Library

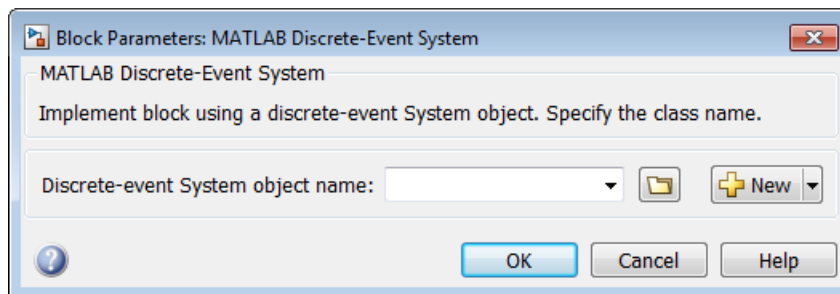
SimEvents Library: User-Defined

## Description



This block allows you to create and author discrete-event systems in MATLAB using the MATLAB System block. With this block, you can author an event-driven entity-flow system using MATLAB, and use it in your Simulink model.

## Dialog Box



### System object name

Specify the full name of the user-defined discrete-event System object class without the file extension. This entry is case sensitive. The class name must exist on the MATLAB path.

You can specify a discrete-event System object name in one of these ways:

- Enter the name in the text box.
- Click the list arrow attached to the text box. If valid System objects exist in the current folder, the names appear in the list. Select a System object from this list.
- Browse to a folder that contains a valid discrete-event System object. If the folder is not on your MATLAB path, the software prompts you to add it.

If you need to create a discrete-event System object, you can create one from a template by clicking **New**.

After you save the SimEvents System object, you can enter the name in the **System object name** text box.

Use the full name of the user-defined discrete-event System object class name. The block does not accept a MATLAB variable that you have assigned to a discrete-event System object class name.

### **New**

Click this button to create a SimEvents System object from a template.

Select one of these options.

- **Basic**

Starts MATLAB Editor and displays a template for a simple discrete-event System object using the fewest System object methods.

After you save the SimEvents System object, you can enter the name in the **System object name** text box.

### **See Also**

Discrete Event Chart | Entity Generator

### **Related Examples**

- Simevents Examples

### **More About**

- “Discrete-Event Systems Created with a System Object”
- “SimEvents Common Design Patterns”

**Introduced in R2016a**

# Entity Gate

Gate entities

## Library

SimEvents Library



## Description

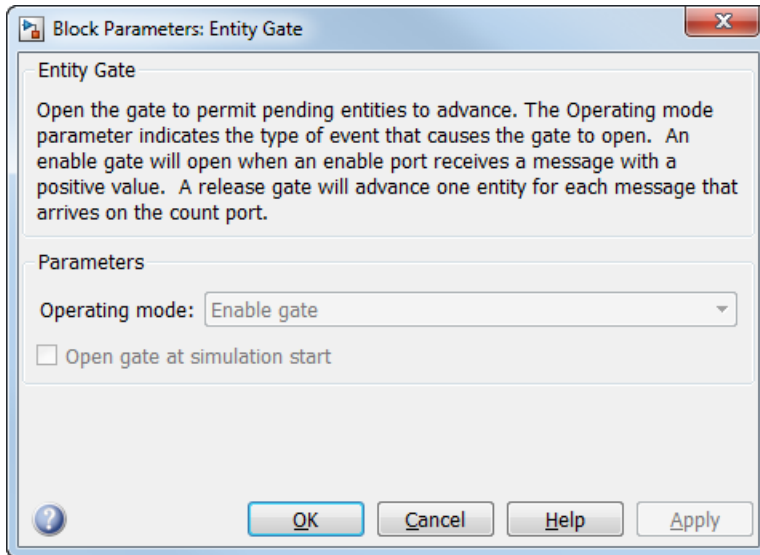
When the **Operating mode** parameter is set to **Enable gate**, this block represents a gate that is open whenever the control signal at the input port is positive, and closed whenever the signal is zero or negative. By definition, an open gate permits entity arrivals as long as the entities would be able to advance immediately to the next block, while a closed gate forbids entity arrivals. The input signal is a numerical signal of type double. Because the signal can remain positive for a time interval of arbitrary length, an enabled gate can remain open for a time interval of arbitrary length. The length can be zero or a positive number.

When the **Operating mode** parameter is set to **Release gate**, this block permits the arrival of one pending entity when a signal-based event or function call occurs; at all other times, the entity input port of the block is unavailable. By definition, the opening of the gate permits one pending entity to arrive if the entity is able to advance immediately to the next block.

No simulation time passes between the opening and subsequent closing of the gate. The gate opens and then closes in the same time instant. If no entity is already pending when the gate opens, then the gate closes without processing any entities.

The **Open gate at simulation start** parameter opens the gate at the start of simulation.

## Dialog Box



### Operating mode

Select the mode of operation of this gate. Select **Enable gate** to allow entity arrivals upon receiving a message with a positive value. Select **Release gate** to allow one pending entity arrival per message arrival.

### Open gate at simulation start

Select this option to open the gate at the start of the simulation.

### See Also

Composite Entity Creator | Composite Entity Splitter | Discrete Event Chart | Entity Server | Entity Input Switch | Entity Multicast | Entity Output Switch | Entity Queue | Entity Replicator

### Related Examples

- Simevents Examples

### More About

- “SimEvents Common Design Patterns”



**Introduced in R2016a**

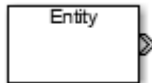
## Entity Generator

Generate entities

### Library

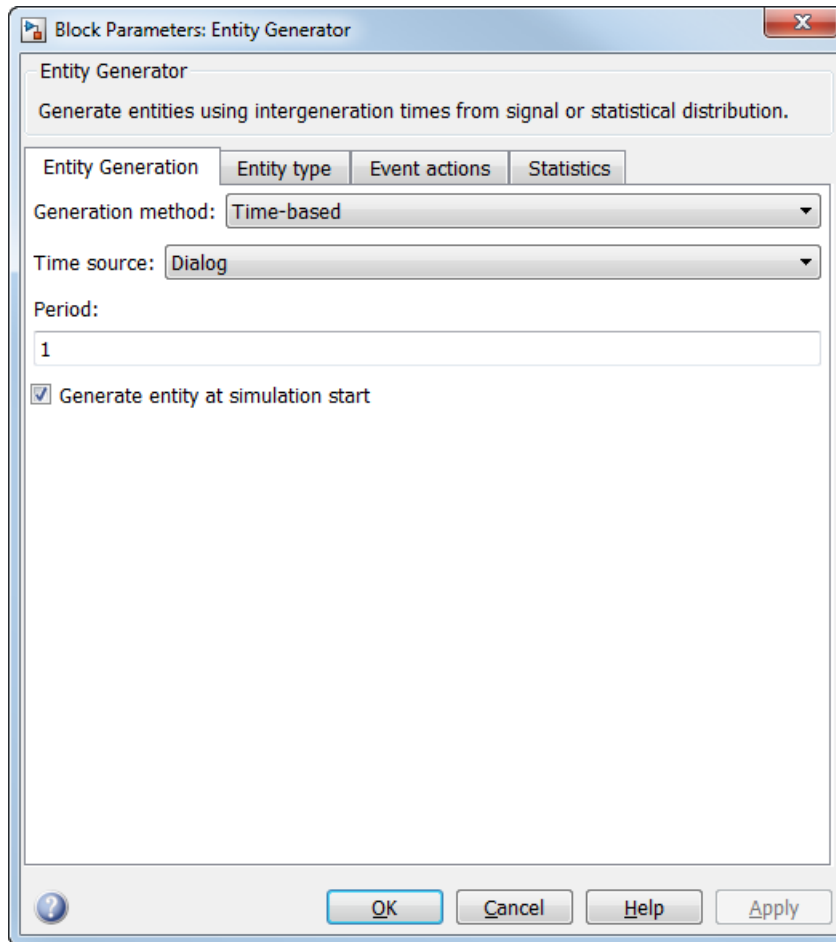
SimEvents Library: Basic

### Description



This block generates entities.

## Dialog Box



### Generation method

Select the method of generation. Choose **Time-based** to generate entities using intergeneration times from an input signal or statistical distribution. Choose **Event-based** for an external event to determine the entity intergeneration time.

### Time source

Select the source of the intergeneration time. Select **Dialog** to specify a fixed period between entity generations. Select **Signal port** to generate entities based on an input signal. Select **MATLAB action** to define a MATLAB script that defines the intergeneration time. This parameter is visible when **Generation method** is set to **Time-based**.

### **Period**

Define the period between the generation of entities. This parameter is visible when **Generation method** is set to **Time-based**.

### **Generate entity at simulation start**

Check this box to generate an entity at the start of the simulation.

### **Entity type**

Choose the type of entity to generate. The **Anonymous** type has a data value associated with it. The **Structured** type includes attributes that you can set. The **Bus object** type lets you generate bus objects as entities.

### **Entity priority**

Determines the priority of the generated entity. A lower value indicates higher priority.

### **Entity type name**

Specify the name of the generated entity. This parameter is visible when **Entity type** is set to **Bus object** or **Structured**.

### **Data initial value**

Specify the initial value of the data of an anonymous entity. This parameter is visible when you set **Entity type** to **Anonymous**.

### **Define attributes**

Define the attributes of the generated entity. This parameter is visible when **Entity type** is set to **Structured**.

---

**Note:** When done, you can export the structured entity type as a bus object, with the name **Entity type name**, to the base workspace. You need to export this bus object when using the **MATLAB Discrete-Event System** and **Discrete Event Chart** blocks.

---

### **Event actions**

Specify the behavior of the entity on certain events. Define the behavior in the **Event action** parameter. For example, the **Generate** action is called when an entity is generated.

**Event action**

Define the behavior for the event action specified in **Event actions**.

**Number of entities departed, d**

Outputs the number of entities that have departed the block.

**Pending entity present in block, pe**

Indicates whether there are pending entities that have yet to depart the block.

**Average intergeneration time, w**

Outputs the average time between generation of entities.

**See Also**

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

**Related Examples**

- Simevents Examples

**More About**

- “SimEvents Common Design Patterns”

**Introduced in R2016a**

## Entity Terminator

Terminate entities

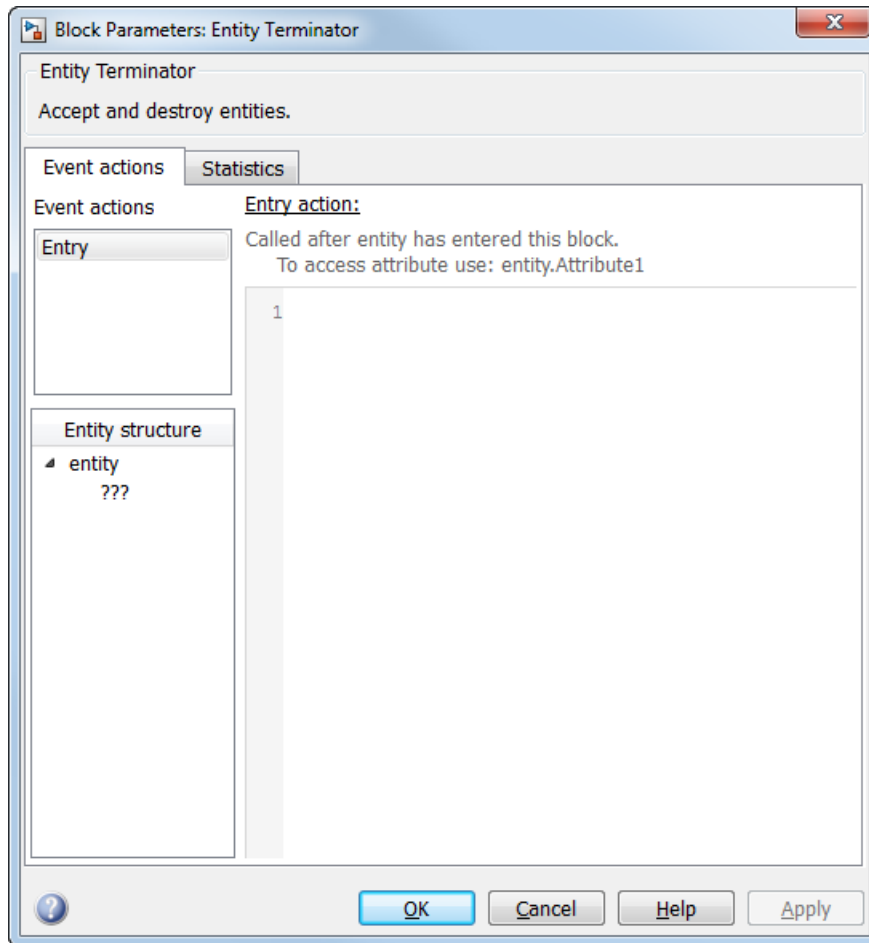
## Library

SimEvents Library: Basic

## Description

This block accepts and destroys entities.

## Dialog Box



### Event actions

Specify the behavior of the entity in certain events. Define the behavior in the **Event action** parameter. For example, the **Generate** action is called after an entity is generated.

### Event action

Define the behavior for the event action specified in **Event actions**.

### **Number of entities arrived, a**

Outputs the number of entities that have arrived at the block.

### **See Also**

Entity Terminator | Entity Generator | Entity Queue

### **Related Examples**

- Simevents Examples

### **More About**

- “SimEvents Common Design Patterns”

**Introduced in R2016a**



# Entity Input Switch

Switch input entities

## Library

SimEvents Library: Routing



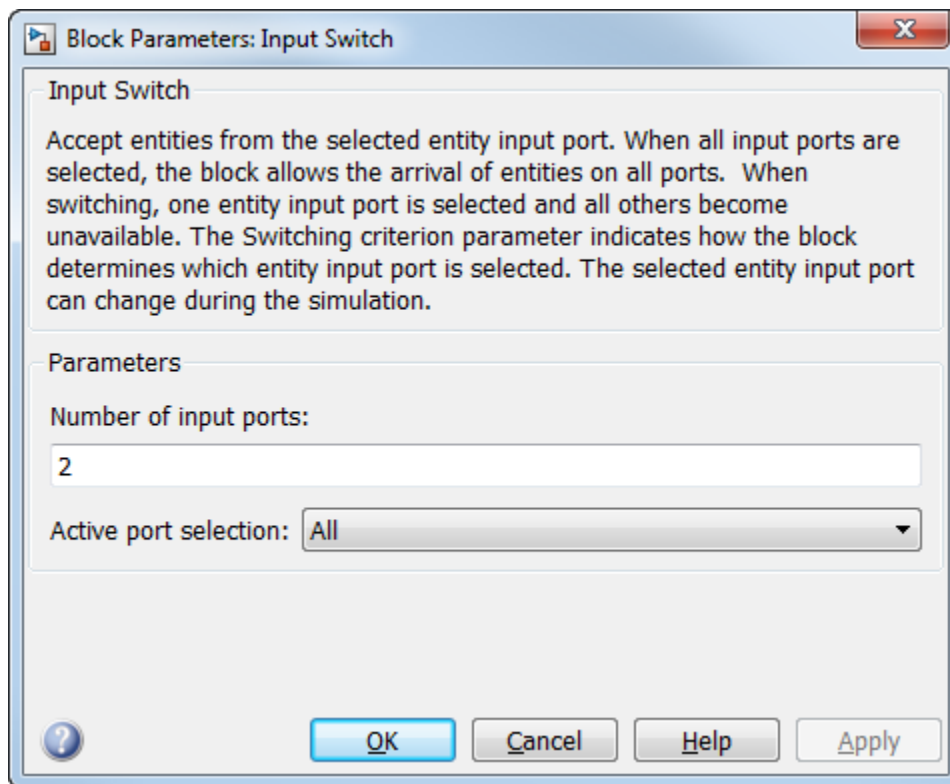
---

**Note:** This page is the block reference page for the Input Switch block introduced in R2016a. To see the documentation on the previous Input Switch block, see [Input Switch \(Obsolete\)](#).

---

This block allows arrival of entities at its ports. The selected entity input port can change during the simulation.

## Dialog Box



### Number of input ports

Determines how many entity input ports the block has.

### Active port selection

Select **All** to allow arrival of entities at all ports. Select **Switch** to allow arrival of an entity at only one port at a time.

### Switching criterion

Select the criterion for switching between input ports. Select **Round robin** to select ports in a round robin fashion. Select **From control port** to let the control port determine the selected port. Select **Equiprobable** to let the block randomly select any port with equal probability.

**Initial port selection**

Specify which port to allow arrival of entity from initially.

**Seed**

Specify the seed for the random number generator to determine the input port. This parameter is visible when **Switching criterion** is set to Equiprobable.

**See Also**

Composite Entity Creator | Composite Entity Splitter | Entity Replicator | Entity Gate | Entity Multicast | Entity Output Switch | Entity Queue | Entity Terminator | Multicast Receive Queue

**Related Examples**

- Simevents Examples

**More About**

- “SimEvents Common Design Patterns”

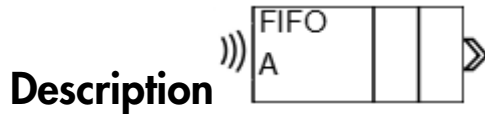
**Introduced in R2016a**

# Multicast Receive Queue

Receive multicast entities

## Library

SimEvents Library: Routing



This block is an Entity Queue block with the **Entity arrival source** parameter set to Multicast. For more information, see the documentation for the Entity Queue block.

## See Also

Entity Generator | Entity Multicast

## Related Examples

- “Broadcast Entities Using Multicast Mode”
- Simevents Examples

## More About

- “SimEvents Common Design Patterns”

Introduced in R2016a

# Entity Multicast

Send multicast entities

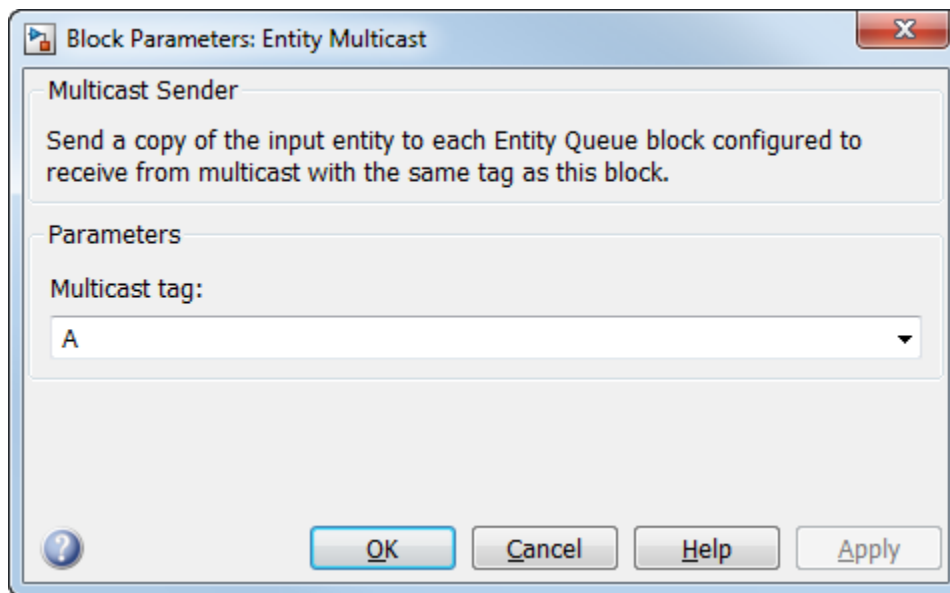
## Library

SimEvents Library: Routing



This block broadcasts entities.

## Dialog Box



**Multicast tag**

Specify the tag with which to broadcast the entities.

### **See Also**

Entity Generator | Multicast Receive Queue

### **Related Examples**

- “Broadcast Entities Using Multicast Mode”
- Simevents Examples

### **More About**

- “SimEvents Common Design Patterns”

**Introduced in R2016a**

# Entity Output Switch

Output entities

## Library

SimEvents Library: Routing



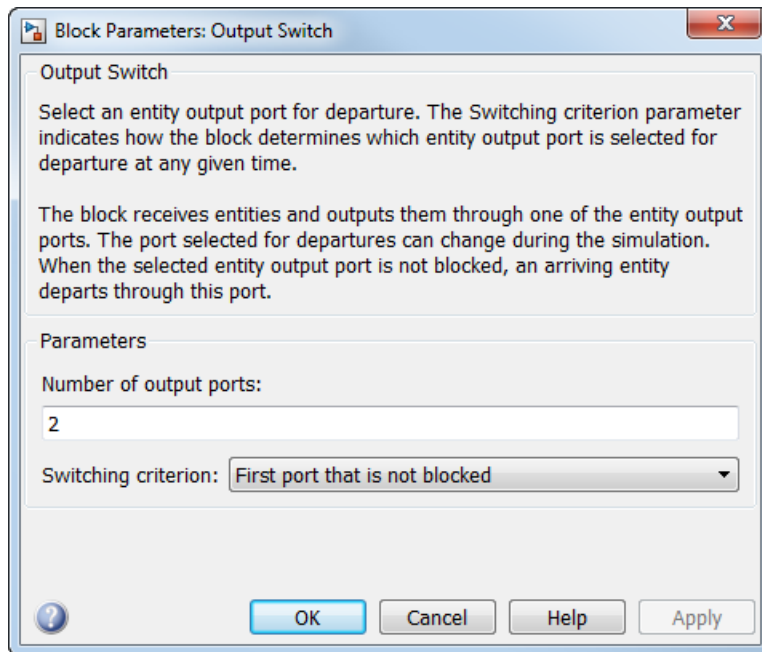
---

**Note:** This page is the block reference page for the Output Switch block introduced in R2016a. To see the documentation for the previous Output Switch block, see [Output Switch \(Obsolete\)](#).

---

The block allows you to select an output port for departure of an entity.

## Dialog Box



### Number of output ports

Specify the number of output ports.

### Switching criterion

Choose the criterion for switching between output ports. Select **First port that is not blocked** to output the entity to the first unblocked port. Select **Round robin** to output entities in a round robin fashion among the output ports. Select **From control port** to introduce a control port that specifies the output port for entity departure. Select **From attribute** to specify an attribute that determines the output port. Select **Equiprobable** to randomly select an output port for entity departure.

### Initial port selection

Specify the output port at the start of the simulation. This parameter is visible when **Switching criterion** is set to **Round robin** or **From control port**.

### Switch Attribute name



Specify the attribute that determines the output port. This parameter is visible when **Switching criterion** is set to `From attribute`.

### **Seed**

Specify the seed for the random number generator to determine the output port. This parameter is visible when **Switching criterion** is set to `Equiprobable`.

### **See Also**

Composite Entity Creator | Composite Entity Splitter | Entity Replicator | Entity Gate | Entity Input Switch | Entity Multicast | Entity Queue | Entity Terminator | Multicast Receive Queue

### **Related Examples**

- Simevents Examples

### **More About**

- “SimEvents Common Design Patterns”

**Introduced in R2016a**

## Entity Queue

Enqueue entities

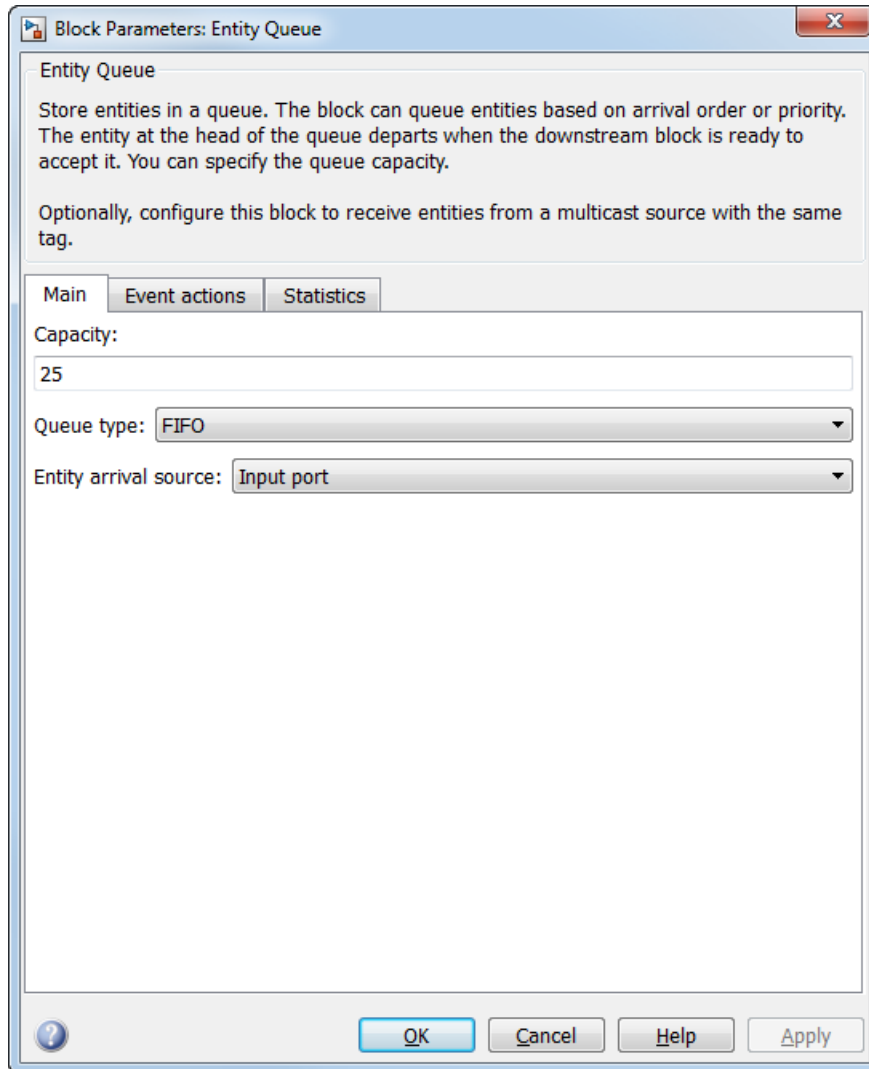
## Library

SimEvents Library: Basic



This block stores entities in a queue, based on order of arrival or priority. The entity at the head of the queue departs when the downstream block is ready to accept it.

## Dialog Box



### Capacity

Specify the capacity of the queue.

### **Queue type**

Choose the type of queue. Select **FIFO** for a first-in-first-out queue. Select **LIFO** for a last-in-first-out queue. Select **Priority** to store the entities in order of their priority.

### **Entity arrival source**

Choose the source of arrival for the entities. Select **Multicast** to receive entities broadcast from **Entity Multicast** blocks.

### **Multicast tag**

Specify the tag when accepting entities broadcast via multicast sources. This parameter is visible when you set **Entity arrival source** to **Multicast**.

### **Priority source**

Specify which attribute of the entity determines its priority. This parameter is visible when you set **Queue type** to **Priority**.

### **Sorting direction**

Choose the direction of sorting entities based on priority.

### **Event actions**

Specify the behavior of the entity in certain events. Define the behavior in the **Event action** parameter. For example, the **Generate** action is called after an entity is generated.

### **Event action**

Define the behavior for the event action specified in **Event actions**.

### **Number of entities departed, d**

Outputs the number of entities that have departed the block.

### **Number of entities in block, n**

Outputs the number of entities present in the block, which have yet to depart.

### **Average wait**

Outputs the average wait time for entities in the block.

### **Average queue length**

Outputs the average length of the entity queue.

### **See Also**

Entity Generator | Entity Multicast | Entity Server | Multicast Receive Queue

## **Related Examples**

- Simevents Examples

## **More About**

- “Queues and Servers”
- “Behavior and Features of Queues”
- “SimEvents Common Design Patterns”

**Introduced in R2016a**

# Entity Replicator

Replicate entities

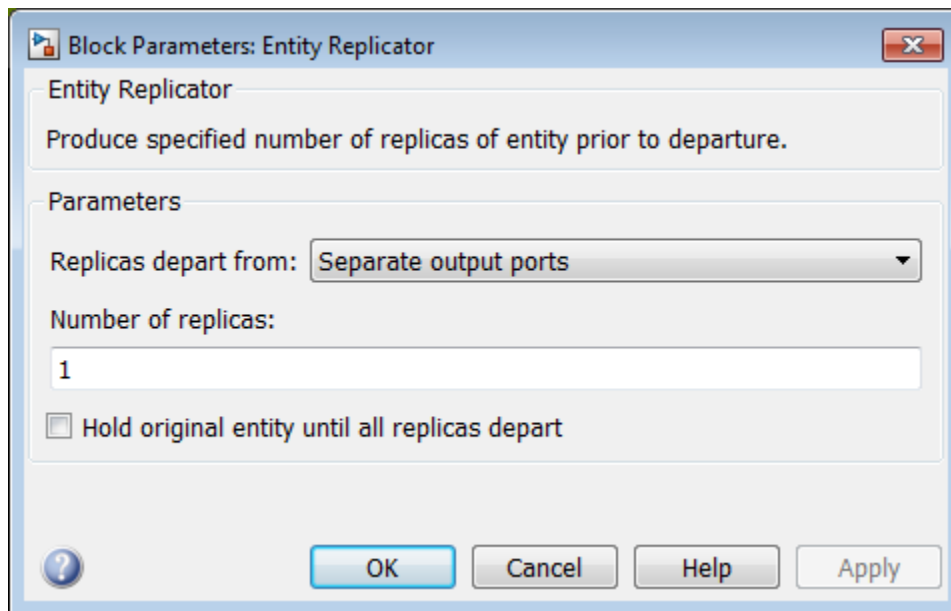
## Library

SimEvents Library: Routing



This block duplicates entities. It outputs the duplicate entities and can also output the original entity. There is always an output port for the original entity.

## Dialog Box



**Replicas depart from**

Choose how replicas depart. Select **Separate output ports** to output one entity from each output port. Select **Single output port** to output all replicas from a single output port.

**Replication amount source**

Choose the source to specify the number of replicas. Select **Dialog** to specify the number of replicas in the dialog box. Select **Attribute** to select an attribute that will specify the number of replicas. This parameter is visible when you set **Port replication mode** to **Single output port**.

**Number of replicas**

Specify the number of replicas. This number depends on the selection of the **Replicas depart from** parameter. If you select **Single output port**, all replicas depart from this output port. If you select **Separate output ports**, each replica has its own port.

**Replicate attribute name**

Specify the attribute that determines the number of replicas. This parameter is visible when you set **Port replication mode** to **Single output port** and **Replication amount source** is set to **Attribute**.

**Hold original entity until all replicas depart**

Select this check box to hold the original entity until all the replicas have departed.

**See Also**

Entity Generator | Entity Server

**Related Examples**

- Simevents Examples

**More About**

- “SimEvents Common Design Patterns”

**Introduced in R2016a**

## Resource Acquirer

Acquire entity resources

## Library

SimEvents Library: Resources



This block defines resources that entities can use during model simulation. Use the **Resource Acquirer** and **Resource Releaser** blocks to work with these resources.

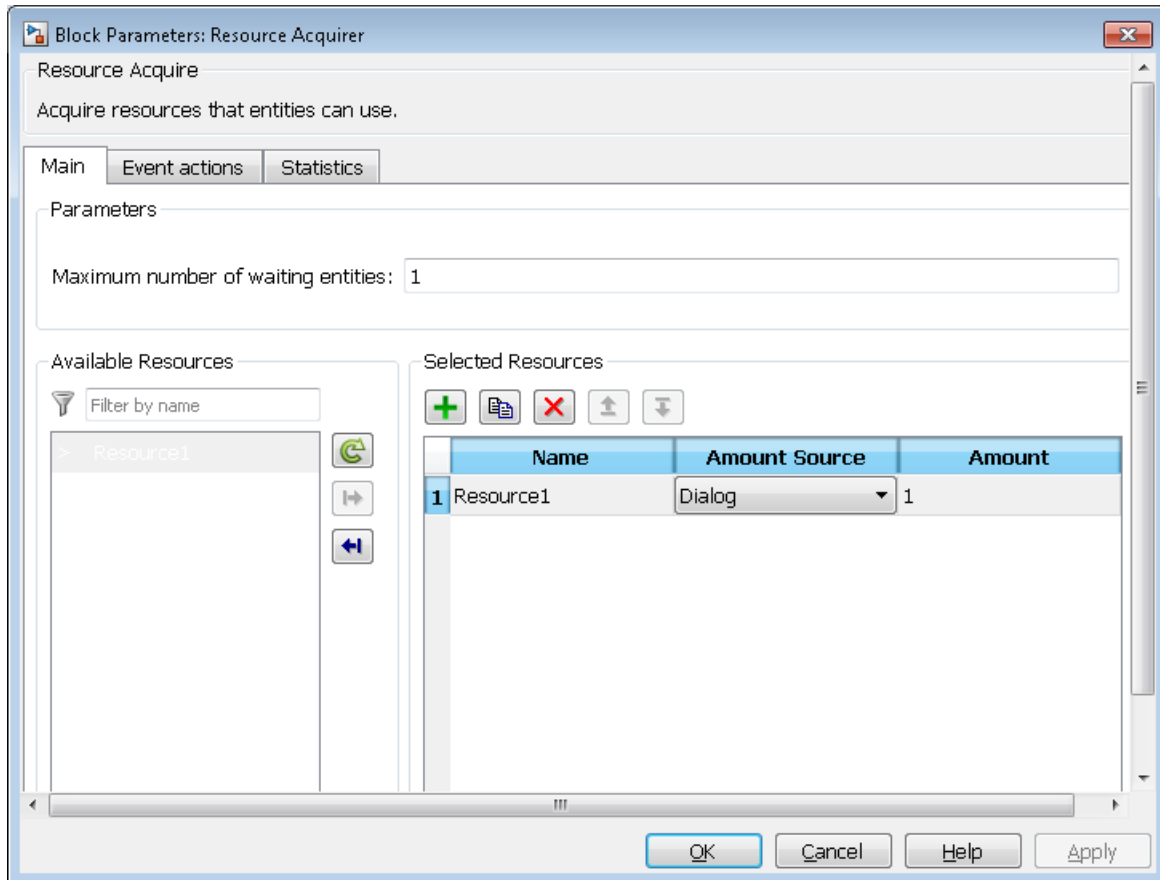
Initialize the block with specified amount of available resources. Then:

- Use one or more **Resource Acquirer** blocks to reserve the use of those resources.
- Use a **Resource Releaser** block to return resources back to this block for future use.

Resources are visible to the current subsystem and its children. Resource are not visible to parent subsystems.



## Dialog Box



### Maximum number of waiting entities

Enter the maximum number of entities that can wait for a resource.

### Event actions

Specify the behavior of the entity on certain events. Define the behavior in the **Event action** parameter. For example, the **Entry** action is called when an entity is generated.

### Event actions

Define the behavior for the event action specified in **Event actions**.

**Number of entities departed**

Select this check box to use the first output port.

**Number of entities in block**

Select this check box to use the second output port.

**Average wait**

Select this check box to use the third output port.

**Available Resources**




Use the **Available Resources** controls to:

- Select the resources from the resources defined in all the **Resource Pool** blocks in the model.
- Add the resources to the **Selected Resources** table, where you can configure resource acquisition details.

The list displays all the available resources in the model. (If there are no resources, the **Available Attributes** list is empty.)

If the resource list is long, you can type the resource name in the text box to filter the list.

Use the buttons in the **Available Resources** section to help build the resources table. The buttons perform these actions.

Button	Action
	Refresh the <b>Available Resources</b> list. The list updates with any upstream model changes you make while the block dialog box is open.
	Add the selected resources to the <b>Selected Resources</b> table.
	Move the selected resource from the <b>Selected Resources</b> table to the <b>Available Resources</b> list.  <b>Note:</b> If the selected resource is one you added manually, this button appears dimmed.

The message area below the available resources list displays additional messages about the resources, as they apply.

Message	Meaning
> Resource already selected	You have already added the resource to the <b>Selected Resources</b> table. You cannot add the resource to the table again.






## Selected Resources

Use the controls under **Selected Resources** to build and manage the list of resources to attach to the entity. Each resource appears as a row in a table.

Using these controls, you can:

- Add a resource manually.
- Modify a resource that you added to the table from the **Available Resources** list to attach to the entity.

The buttons under **Selected Resources** perform these actions:

Button	Action	Notes
	Add a template resource to the table.	Rename the resource and specify its properties.
	Add a copy of the selected resource to the table to use as the basis of a new resource.	Rename the copy. Two resources cannot have the same name.
	Remove the selected resource from the <b>Selected Resources</b> table.	When you delete a resource this way, no confirmation appears and you cannot undo the operation.
	Move the selected resource up in order in the <b>Selected Resources</b> table.	N/A
	Move the selected resource down in order in the <b>Selected Resources</b> table.	N/A

**Note:** If you delete a row and apply the change, the deletion can affect signal output ports corresponding to other attributes. For example, if the block has a signal output port **A2**

and you delete the attribute with a port marked **A1**, the block relabels **A2** as **A1**. Verify that any signal that connects to the relabeled port is still connected as you expect.

---

Property	Specify	Use
<b>Name</b>	The name of the resource. Each resource must have a unique name.	Double-click the existing name, and then type the new name.
<b>Amount Source</b>	Whether the resource amount, that an entity requests, comes from the dialog box or an attribute.	Select <b>Dialog</b> or <b>Attribute</b> . If you select <b>Attribute</b> , the source of the resource amount comes from the attribute of the entity. This option allows each entity to acquire varying amounts of resources. For more information, see “Set Resource Amount with Attributes”
<b>Amount</b>	The value to assign to the resource (when the resource comes from the dialog box).	Double-click the value, and then type the value you want to assign.  This value is the number of resources acquired per entity. For example, if <b>Amount</b> is <b>3</b> , each entity that arrives at the <b>Release Acquire</b> must wait to acquire 3 resources before departing the block.

### See Also

Entity Generator | Resource Pool | Resource Releaser

### Related Examples

- “Model with Resources”

- Simevents Examples

## **More About**

- “SimEvents Common Design Patterns”

**Introduced in R2016a**

## Resource Pool

Pool entity resources

## Library

SimEvents Library: Resources

## Description



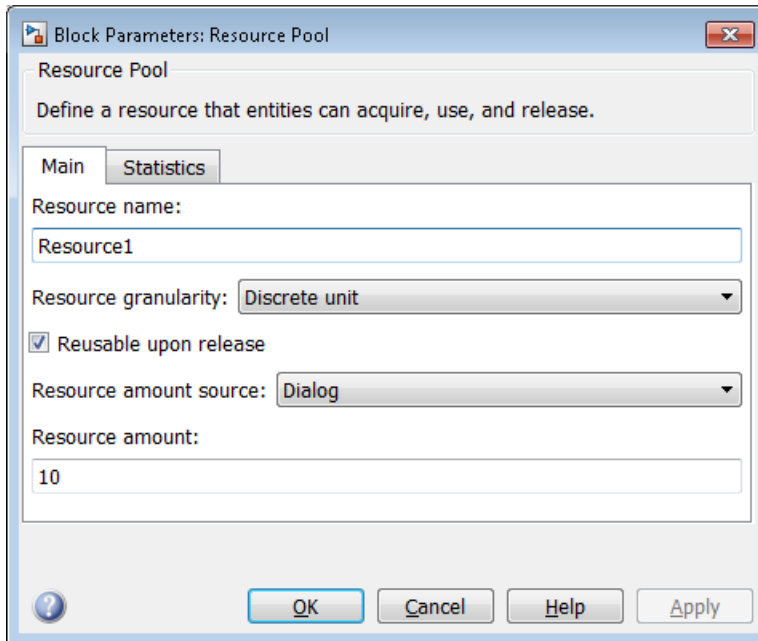
This block defines resources that entities can use during model simulation. Use the **Resource Acquirer** and **Resource Releaser** blocks to work with these resources.

Initialize the block with the specified amount of available resources. Then:

- Use one or more **Resource Acquirer** blocks to reserve the use of those resources.
- Use a **Resource Releaser** block to return resources back to this block for future use.

Resources are visible to the current subsystem and its children. Resource are not visible to parent subsystems.

## Dialog Box



### Resource name

Enter name of entity resource.

### Resource granularity

Select granularity of resource use.

- **Discrete unit** — Use whole number increment.
- **Fractional amount** — Use fractional increment.

### Reusable upon release

- Select this check box to allow this resource to return to the resource pool upon release. An example of such a resource is a table in a restaurant, which is available for reuse when a customer leaves.

Selecting this check box enables the **Resource amount source** check box.

- Clear this check box to prevent this resource from returning to the resource pool upon release. An example of such a resource is food in a restaurant, which is not reusable upon consumption.

### **Resource amount source**

Select resource amount source.

- Dialog

Selecting this option enables the `Resource amount` parameter.

### **Resource amount**

Enter amount of resource.

### **Amount in use**

Select this check box to use the first output port.

### **Average utilization**

Select this check box to use the second output port.

## **See Also**

Entity Generator | Resource Releaser | Resource Acquirer

## **Related Examples**

- “Model with Resources”
- Simevents Examples

## **More About**

- “SimEvents Common Design Patterns”

**Introduced in R2016a**



# Resource Releaser

Release entity resources

## Library

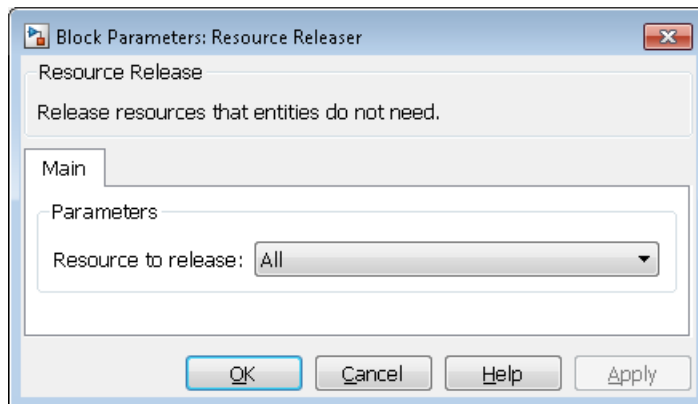
SimEvents Library: Resources



## Description

This block releases the use of resources for a passing entity. You can specify that the block release certain resource types or release all resources.

## Dialog Box



### Resource to release

Select the resources to release.

- All

Release the use of all resources for a passing entity.

- Selected

Release selected resources. Selecting this option enables the **Available Resources** table.

## Available Resources




Use the **Available Resources** controls to:

- Select the resources from the resources defined in all the **Resource Pool** blocks in the model
- Add the resources to the **Selected Resources** table, where you can modify them.

The list displays all the resources in the model. (If there are no resources, the **Available Resources** list is empty).

If the resource list is long, you can type the resource name in the text box to filter the list.

Use the buttons in the **Available Resources** section to help build the resources table. The buttons perform these actions.

Button	Action
	Refresh the <b>Available Resources</b> list. The list updates with any upstream model changes you make while the block dialog box is open.
	Add the selected resources to the <b>Selected Resources</b> table.
	Move the selected resource from the <b>Selected Resources</b> table to the <b>Available Resources</b> list.  Note: If the selected resource is one you added manually, this button appears dimmed.

The message area below the available resources list displays additional messages about the resources, as they apply.

Message	Meaning
> Resource already selected	You have already added the resource to the <b>Selected Resources</b> table. You cannot add the resource to the table again.






## Selected Resources

Use the controls under **Selected Resources** to build and manage the list of resources to release. Each resource appears as a row in a table.

Using these controls, you can:

- Add a resource manually.
- Modify a resource that you added to the table from the **Available Resources** list to release.

The buttons under **Selected Resources** perform these actions.

Button	Action	Notes
	Add a template resource to the table.	Rename the resource and specify its properties.
	Add a copy of the selected resource to the table to use as the basis of a new resource.	Rename the copy. Two resources cannot have the same name.
	Remove the selected resource from the <b>Selected Resources</b> table.	When you delete a resource this way, no confirmation appears and you cannot undo the operation.
	Move the selected resource up in order in the <b>Selected Resources</b> table.	N/A
	Move the selected resource down in order in the <b>Selected Resources</b> table.	N/A

**Note:** If you delete a row and apply the change, the deletion can affect signal output ports corresponding to other attributes. For example, if the block has a signal output port **A2**

and you delete the attribute with a port marked **A1**, the block relabels **A2** as **A1**. Verify that any signal that connects to the relabeled port is still connected as you expect.

---

Property	Specify	Use
Name	The name of the resource to release.	Double-click the existing name, and then type the new name.

### See Also

Entity Generator | Resource Acquirer | Resource Pool

### Related Examples

- “Model with Resources”
- Simevents Examples

### More About

- “SimEvents Common Design Patterns”

**Introduced in R2016a**

# Entity Server

Serve entities

## Library

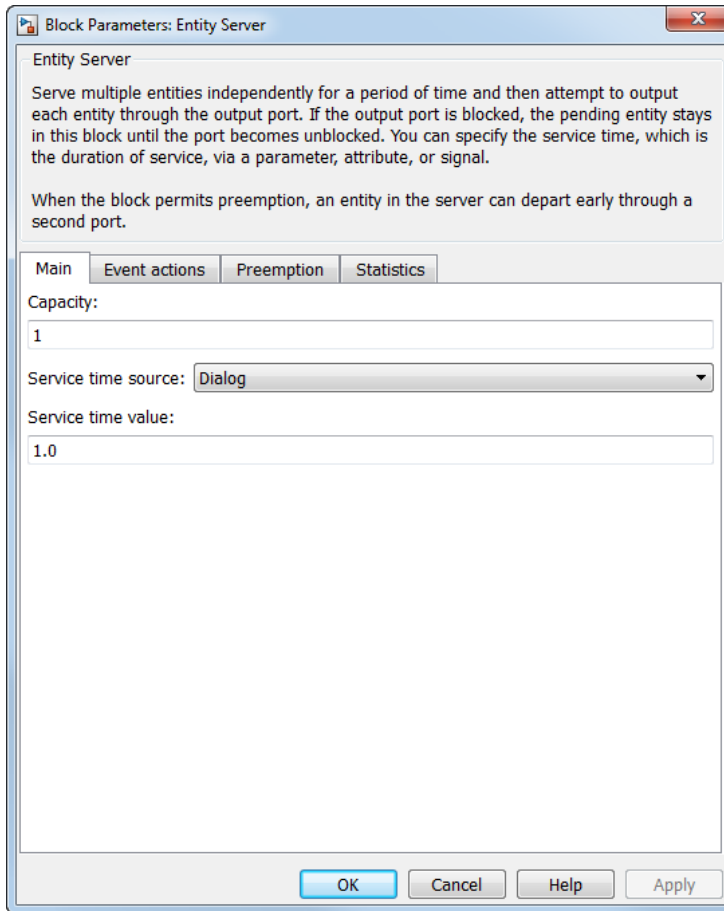
SimEvents Library: Basic

## Description



This block serves entities as they arrive. It can serve multiple entities simultaneously and output each entity through the output port, unless the port is blocked. When the block permits preemption, an entity in the server can depart early through a second port.

## Dialog Box



### Capacity

Specify the number of entities the block can serve simultaneously.

### Service time source

Choose the source to specify the service time.

### Service time attribute name

Specify the attribute used to determine the service time. This parameter is visible when **Service time source** is set to **Attribute**.

**Service time value**

Specify the value of the service time. This parameter is visible when **Service time source** is set to **Dialog**.

**Service time action**

Use MATLAB code to specify service time.

**Event actions**

Specify the behavior of the entity in certain events. Define the behavior in the **Event action** parameter. For example, the **Generate** action is called after an entity is generated.

**Event action**

Define the behavior for the event action specified in **Event actions**.

**Permit preemption based on attribute**

Select this check box if you want to allow preemption of entities. Selecting this check box enables these parameters:

- **Sorting attribute name**
- **Sorting direction**
- **Write residual time to attribute**
- **Number of pending entities, np**

**Sorting attribute name**

Specify the name of the attribute used to determine the priority.

**Sorting direction**

Choose the direction of sorting the entities.

**Write residual time to attribute**

Allow the block to save the residual service time from a preempted entity to an attribute.

This parameter is visible when the **Permit preemption based on attribute** parameter is selected. Selecting this parameter enables the **Residual time attribute name** parameter.

**Residual time attribute name**

Specify the name of the attribute to contain the residual service time of a preempted entity. This parameter is visible when the **Write residual time to attribute** parameter is selected.

**Number of entities departed, d**

Outputs the number of entities that have departed the block.

**Number of entities in block, n**

Outputs the number of entities present in the block.

**Pending entity present in block, pe**

Indicates whether there are entities present in the block that have yet to depart.

**Number of pending entities, np**

Outputs the number of pending entities in the block, which have yet to depart.

**Average wait**

Outputs the average wait time for entities in the block.

**Utilization**

Outputs the average number of entities being served.

**Number of entities preempted, np**

Outputs the number of preempted entities. This check box appears if the **Permit preemption based on attribute** check box is selected.

**See Also**

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

**Related Examples**

- Simevents Examples

**More About**

- “Queues and Servers”
- “Behavior and Features of Servers”
- “SimEvents Common Design Patterns”

**Introduced in R2016a**



# Message Receive

Extract data from received messages

## Library

SimEvents Library: Other

## Description

The **Message Receive** block extracts data from received messages and writes them to the output signal port. If there are no new messages when the block executes, the block uses the **Value source when queue is empty** value:

- Hold last value

Hold data read from the last message.

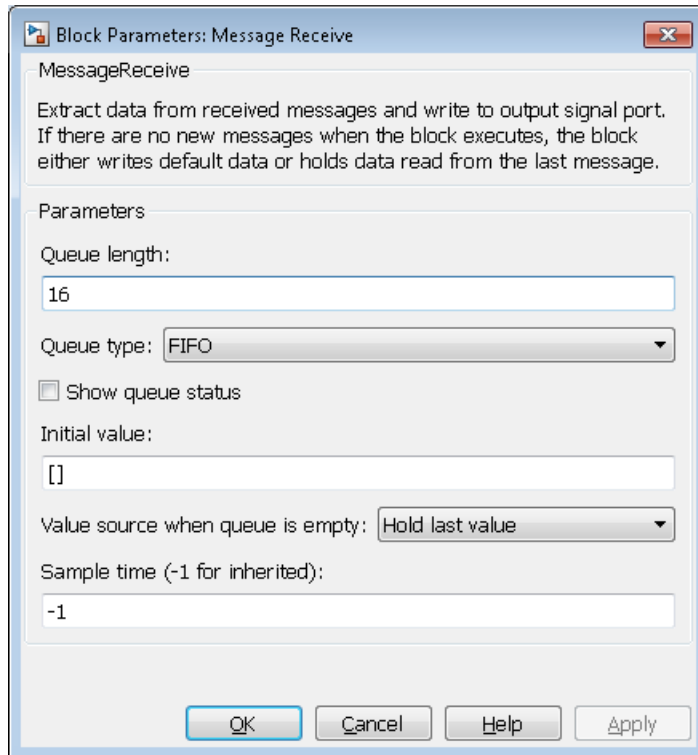
- Use initial value

Writes default data.

## Data Type Support

The **Message Receive** block accepts inputs of any type that Simulink supports, including enumerated types. For more information, see “Data Types Supported by Simulink”.

### Dialog Box



#### Queue length

Specify message queue length.

#### Queue type

Specify message queue type. Selecting **Queue type** > **Priority** enables the **Priority** order parameter.

Choose from:

- FIFO
- LIFO
- Priority

#### Priority order

Specify message queue priority.

Choose from:

- Ascending
- Descending

### **Show queue status**

Select to show queue status.

### **Initial value**

Enter an initial value.

### **Value source when queue is empty**

Specify the value to receive when received message queue is empty.

Choose from:

- Hold last value  
Holds data read from the last message.
- Use initial value  
Writes default data.

### **Sample time (-1 for inherited)**

Specify the time interval between samples.

To inherit the sample time, set this parameter to -1.

See “Specify Sample Time” for more information.

## **See Also**

Message Send

## **Related Examples**

- Simevents Examples

## **More About**

- “SimEvents Common Design Patterns”

**Introduced in R2016a**

# Message Send

Create and send message

## Library

SimEvents Library: Other

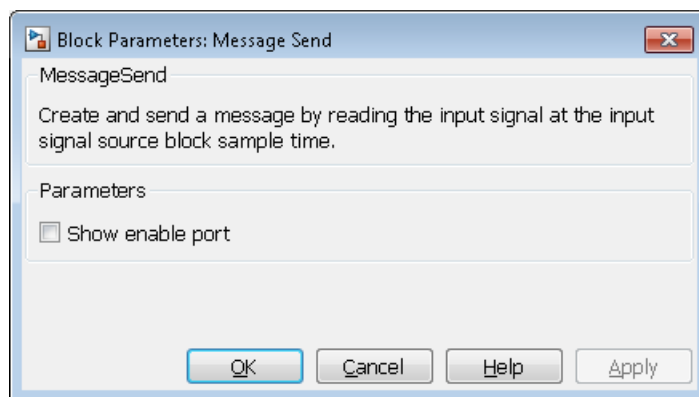
## Description

The Message Send block reads the input signal and creates and sends a message from this signal.

## Data Type Support

The Message Send block accepts inputs of any type that Simulink supports, including enumerated types. For more information, see “Data Types Supported by Simulink”.

## Dialog Box



### **Show enable port**

Select this check box to display the enable port.

### **See Also**

Message Receive

### **Related Examples**

- Simevents Examples

### **More About**

- “SimEvents Common Design Patterns”

**Introduced in R2016a**


# SimEvents Debugger

Debug SimEvents models

## Library

SimEvents Library

## Description

A small rectangular icon with a thin black border. Inside the rectangle, the text "% SimEvents" is on the top line and "% Debugger" is on the bottom line, both in a green, monospace-style font.

The `SimEvents Debugger` block enables the debugger for your SimEvents model. You can:

- Inspect entities and their attribute values in storage blocks
- Set breakpoints on blocks and events
- Watch entities

---

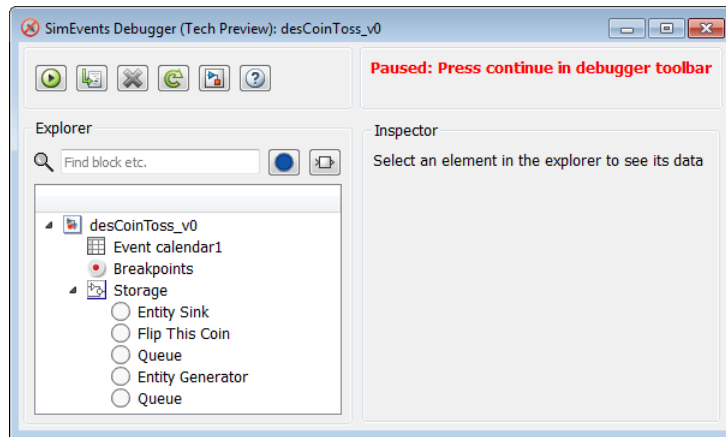
**Note:** The SimEvents debugger is a preliminary version.


---

To start debugging your model:

- 1 From the SimEvents Library, drag the `SimEvents Debugger` block into your SimEvents model.
- 2 In the Simulink editor, click the **Step Forward** button.


The debugger interface appears.




- Click the **Continue simulation** button () to begin the simulation in the debugger.
- To control the simulation of the model, click the buttons.
- To explore their data and behavior, the model tree displays in the left pane. Select the elements in the tree
- When done, on the Simulink editor, click the **Stop** button to stop the simulation.

### Inspect Entities

To inspect entities in the debugger;

- To step to the next time step and inspect entities, in the Simulink editor, click the **Step Forward** button again. This action skips over all events at  $t_{now}$ .
- To step to the next event and inspect entities, in the debugger, click .
- To set a breakpoint:
  - At a particular time, use the Simulation Stepper to set breakpoints.
  - At an event on the event calendar, in the debugger, in the left pane, click an event calendar.



- At every event, in the debugger, in the left pane, select the event calendar. In the Event Calendar Events pane, select the **Break before event execution** check box.
- When an entity enters a block, in the debugger, select the block. At the bottom of the Inspector pane, select the **Break upon entry** check box.
- When an entity leaves a block, in the debugger, select the block. At the bottom of the Inspector pane, select the **Break prior to entity exit** check box.
- To go to a breakpoint:
  - In the debugger, click the **Continue** button (  ).

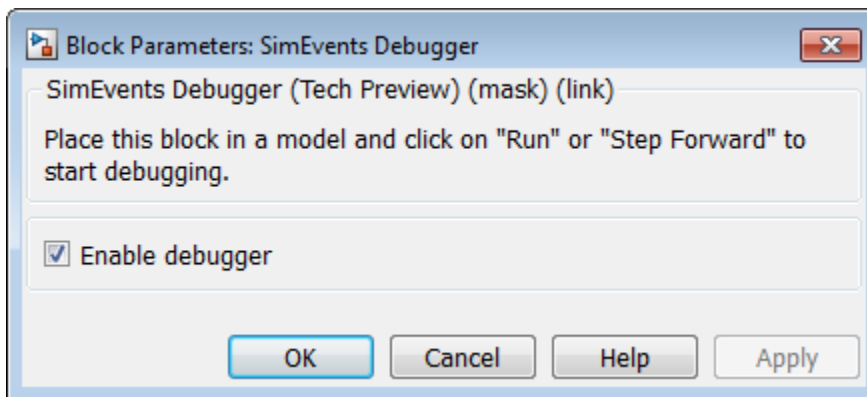
---

**Note:** When you stop the debugger at a breakpoint, the Simulink editor and the MATLAB Command Window appear unresponsive. However, you can inspect entities, set new breakpoints, and continue the simulation from the debugger window.

---

- To watch entities:
  - In the left tree, click the block. In the Inspector pane, select the check box of the entity you want to watch.

## Dialog Box



### Enable debugger

Select this check box to enable the debugger for your model.

## **See Also**

- “Simulation Stepper”

**Introduced in R2016a**

# Entity Sink

Accept or block entities

## Library

SimEvents Sinks

## Description



This block provides a way to terminate an entity path:

- If you select **Input port available for entity arrivals**, the block always accepts entity arrivals.
- If you do not select **Input port available for entity arrivals**, the block never accepts entity arrivals. The simulation issues an error message if an entity attempts to arrive at the block.

## Ports

### Entity Input Ports

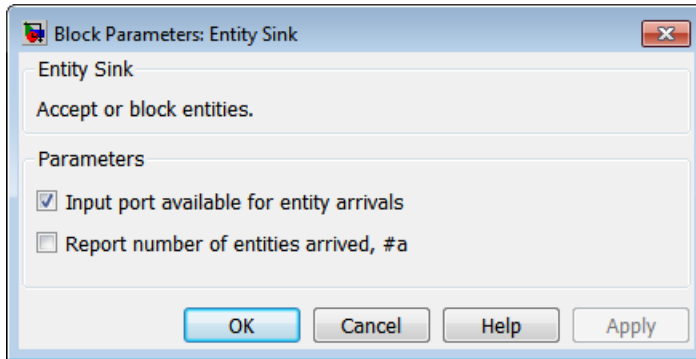
Label	Description
IN	Port for entities that arrive or attempt to arrive.

### Signal Output Ports

Label	Description	Time of Update When Statistic Is On
#a	Number of entities that the block has accepted. You see this port only if you select <b>Input port available for entity arrivals</b> , and then select the <b>Report number of entities arrived, #a</b> check box.	After entity arrival

The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

### Dialog Box



#### **Input port available for entity arrivals**

Determines whether the block accepts or blocks entities that attempt to arrive.

#### **Report number of entities arrived, #a**

Allows you to use the signal output port labeled **#a**. You see this field only if you select **Input port available for entity arrivals**.

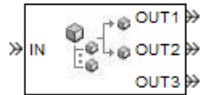
#### **Introduced before R2006a**

# Entity Splitter (Obsolete)

Divide composite entity into component entities

## Library

Entity Management



## Description

This block divides a composite entity into its components and outputs the component entities through each entity output port that is not blocked. A composite entity is an entity that the **Entity Combiner (Obsolete)** block creates using the **Retain structure in departing entity** option. In a typical pairing, the number of entity input ports of the **Entity Combiner** block equals the number of entity output ports of the **Entity Splitter** block.

Timeout events, if any, corresponding to the composite entity are canceled during the splitting operation.

---

**Note:** If you want identical copies of an arriving entity to advance along multiple entity paths, use the **Replicate (Obsolete)** block instead of the **Entity Splitter** block. The **Replicate** block copies entities without regard to their structure.

---

## Attributes and Timers

Attributes and timers from the original component entities (that combined to form the composite entity) are present in the component entities that depart from this block. The values of the attributes and timers might have changed between the combining and splitting operations.

If the composite entity acquired a new attribute or a new timer between the combining and splitting operations, then it is not present in the component entities that depart from this block.

## Complete or Partial Splitting

The **Split entity when** parameter affects the circumstances under which the block accepts an entity to split. Choices are in the table.

Parameter Value	Description
All entity output ports are not blocked	The block accepts an entity to split only when all component entities would be able to depart immediately.
Any entity output port is not blocked	The block accepts an entity to split when at least one component entity would be able to depart immediately.

## Departure of Component Entities

Each time the block splits an entity, the component entities depart in a sequence whose start is determined by the **Departure port precedence** parameter. Choices are in the next table.

Parameter Value	Description	Example
OUT1 port	Each time the block splits an entity, the component entities depart via entity output ports <b>OUT1, OUT2, OUT3,...</b> , in that sequence.	The sequence of departures is always <b>OUT1, OUT2, OUT3,...</b> throughout the simulation.
Round robin	Each time the block splits an entity, the first component entity departs via the port after the one that received preference on the last such occasion. The remaining component entities depart via the subsequent ports in turn.	On a block with three entity output ports, the first time the block splits an entity, the component entities depart in the sequence <b>OUT1, OUT2, OUT3</b> . The second time, the component entities depart in the sequence <b>OUT2, OUT3, OUT1</b> . The third time, the component entities depart in the sequence <b>OUT3, OUT1, OUT2</b> . The fourth time is analogous to the first time, and so on.
Equiprobable	Each time the block splits an entity, the first component entity departs via a randomly selected entity output port. All entity output ports are equally likely to be selected and the <b>Initial seed</b> parameter initializes the random number generation process. The remaining component entities depart via the subsequent ports in turn.	On a block with four entity output ports, if the random number is three, then the component entities depart in the sequence <b>OUT3, OUT4, OUT1, OUT2</b> . If the random number is two on the next such occasion, then the component entities depart in the sequence <b>OUT2, OUT3, OUT4, OUT1</b> .

An example in which the choice of **Departure port precedence** parameter is relevant is a model that sets an attribute on each component entity based on its departure port



and then advances all component entities along a merged path to a **FIFO Queue** block. At each splitting occurrence during the simulation, the **Departure port precedence** parameter determines the sequence of the component entities in the queue.

In some cases, a departure through one entity output port causes another entity output port to become newly blocked. For example, this could happen if two entity output ports connect to a **Path Combiner** block, which in turn connects to a **Single Server** block whose service time is nonzero. Use the **If an output port becomes blocked during split** parameter to determine how the block responds. Choices are in the table below.

Parameter Value	Description
Discard entity	The block discards the component entity that is supposed to depart through the newly blocked entity output port.
Warn and discard entity	The block issues a warning message in the MATLAB Command Window, and discards the component entity that is supposed to depart through the newly blocked entity output port.
Error	The simulation halts with an error message.

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities, which must be composite entities created by the <b>Entity Combiner</b> block using the <b>Retain structure in departing entity</b> option.

### Entity Output Ports

Label	Description
OUT1, OUT2, OUT3, and so on	Entity ports through which component entities depart. The entity that departs via the <b>OUTW</b> port corresponds to the entity that arrived at the <b>INW</b> entity input port of the corresponding <b>Entity Combiner</b> block. The <b>Number of entity output ports</b> parameter determines how many of these entity output ports the block has.

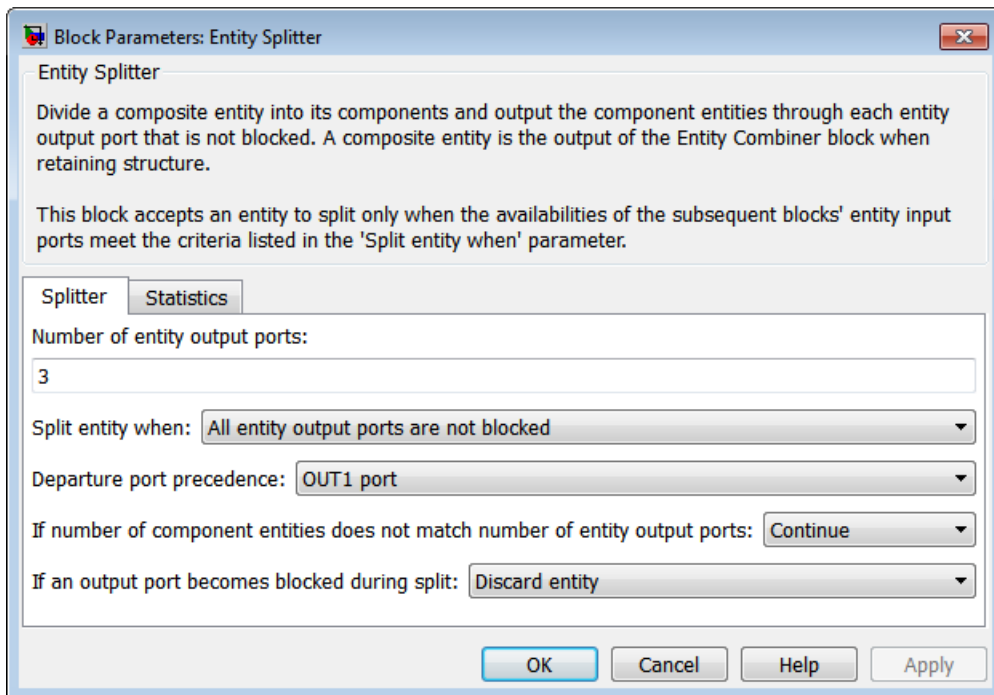
### Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update
#a	Number of entities that have arrived at this block since the start of the simulation.	After entity arrival	1
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	2

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

### Entity Splitter Tab



**Number of entity output ports**

Determines how many entity output ports the block has.

**Split entity when**

Determines whether the block is available to arriving entities whenever at least one entity output port is not blocked, or only when all entity output ports are not blocked.

**Departure port precedence**

Determines the start of the sequence in which the block outputs the component entities, each time the block splits an entity.

**Initial seed**

A nonnegative integer that initializes the random number generator used to determine the output sequence. You see this field only if you set **Departure port precedence** to Equiprobable.

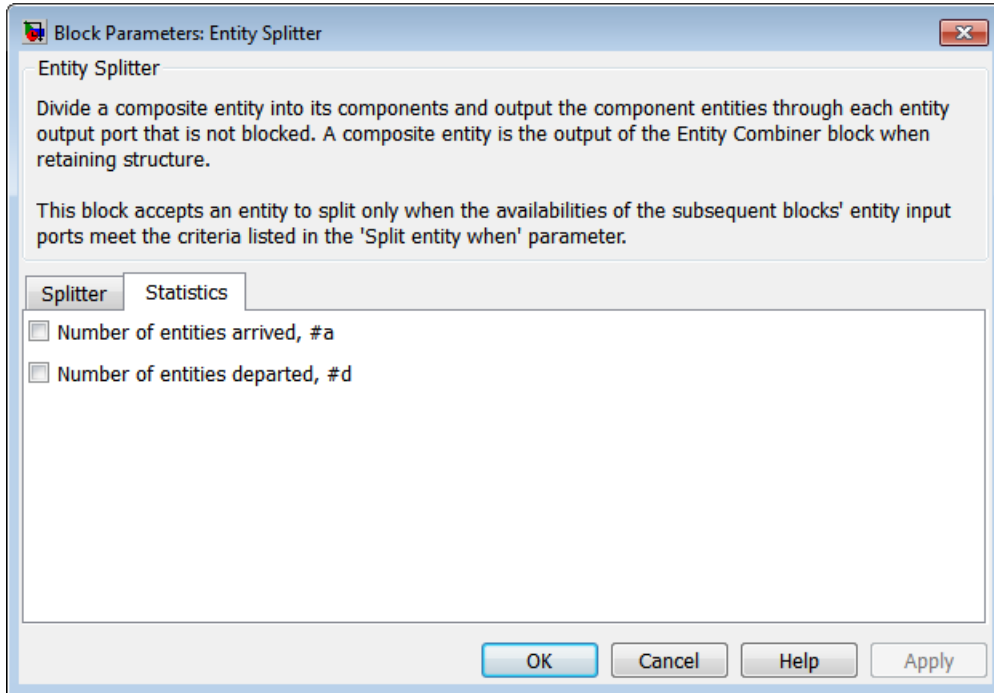
**If number of component entities does not match number of entity output ports**

Determines whether the block issues a message when the number of component entities in the arriving composite entity does not equal the number of entity output ports of this block. “Continue” means that the block ignores any extra entity output ports and discards any extra component entities.

**If an output port becomes blocked during split**

Determines whether the block issues a message when a component entity is unable to depart because an output port becomes blocked during the splitting process. You see this field only if you set **Split entity when** to All entity output ports are not blocked.

## Statistics Tab



### Number of entities arrived

Allows you to use the signal output port labeled **#a**.

### Number of entities departed

Allows you to use the signal output port labeled **#d**.

## Examples

See “Manage Data in Composite Entities”.

## See Also

Entity Combiner (Obsolete)

“Combine Entities”

**Introduced in R2007a**

## Entity-Based Function-Call Event Generator (Obsolete)

Generate function call events corresponding to entities

### Library

Generators / Event Generators




---

**Note:** The Entity-Based Function-Call Event Generator block will be removed in a future release. Use the Entity Departure Function-Call Generator (Obsolete) block instead.

---

This block generates a function call corresponding to each entity that arrives at the block. You can choose whether the block generates the function call before or after the departure. You can use the function call to invoke function-call subsystems, Stateflow blocks, or other blocks that accept function-call inputs.

This block is similar to the Entity Departure Function-Call Generator (Obsolete) block, which offers more flexibility.

### Ports

#### Entity Input Ports

Label	Description
IN	Port for arriving entities.

#### Entity Output Ports

Label	Description
OUT	Port for departing entities.

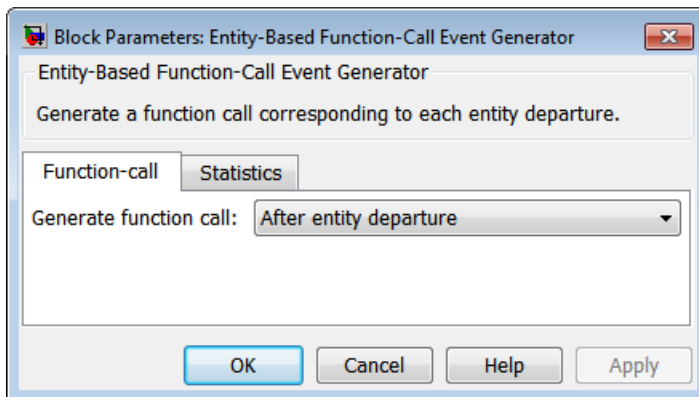
### Signal Output Ports

Label	Description	Time of Update When Port Is Present	Order of Update
f1	Function-call signal.	Before or after entity departure, depending on <b>Generate function call</b> parameter	1
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	3
#f1	Number of function calls the block has generated since the start of the simulation.	After entity departure	2

The initial output value for the numerical signals, which is in effect from the start of the simulation until the first update by the block, is 0.

## Dialog Box

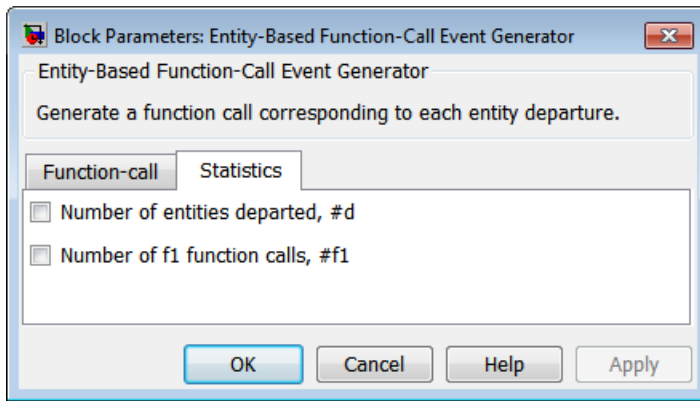
### Function Call Tab



### Generate function call

Determines whether the function call occurs before or after the entity departs from this block.

### Statistics Tab



#### Number of entities departed

Allows you to use the signal output port labeled **#d**.

#### Number of f1 function calls

Allows you to use the signal output port labeled **#f1**.

### See Also

Entity Departure Function-Call Generator (Obsolete), Signal-Based Function-Call Event Generator (Obsolete)

Introduced before R2006a



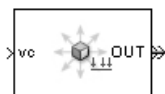
## Event-Based Entity Generator (Obsolete)

Generate entity upon signal-based event or function call

### Library

/

### Description



This block is designed to generate entities when events of a specified type occur.

When to Generate Entities	Generate entities upon Value
Each time the application updates (that is, recomputes and outputs) the value of a signal	Sample time hit from port <code>ts</code>
Each time an input signal has a trigger edge	Trigger from port <code>tr</code>
Each time an input signal changes its value	Change in signal from port <code>vc</code>
Each time an input signal carries a function call	Function call from port <code>fcn</code>

**Note:** An exceptional case is when the block temporarily suspends its normal entity-generation behavior. See the description of the `Delay first pending entity` option in “Responding to Blockage at the Entity Output Port” on page 2-119.

### Responding to Blockage at the Entity Output Port

You can choose how this block responds when the subsequent entity input port is not available to accept the newly generated entity. The responses and corresponding parameters values are in the table.

Response to Blockage	Parameter Values
Error message	Clear the <b>Allow OUT port blocking</b> check box.
The block stores the entity as a pending entity, and immediately discards it. The entity does not depart from the block.	Select <b>Allow OUT port blocking</b> and set <b>Response during blockage period</b> to Discard generated entities
The block stores the entity as a pending entity, and temporarily suspends the generation of additional entities. During this suspension, when the block executes <b>EntityGeneration</b> events, it does not produce new entities. When the subsequent entity input port becomes available, the pending entity departs and the block resumes normal operation.	Select <b>Allow OUT port blocking</b> and set <b>Response during blockage period</b> to Delay first pending entity

## Ports

### Signal Input Ports

Label	Description
<b>ts</b>	When this signal has an update, the block generates an entity. You see this port only if you set <b>Generate entities upon</b> to Sample time hit from port <b>ts</b> .
<b>tr</b>	When this signal satisfies the specified trigger criteria, the block generates an entity. You see this port only if you set <b>Generate entities upon</b> to Trigger from port <b>tr</b> .
<b>vc</b>	When this signal satisfies the specified value-change criteria, the block generates an entity. You see this port only if you set <b>Generate entities upon</b> to Change in signal from port <b>vc</b> .
<b>fcn</b>	When this signal carries a function call, the block generates an entity. This signal must be an event-based function call. You see this port only if you set <b>Generate entities upon</b> to Function call from port <b>fcn</b> .

**Entity Output Ports**

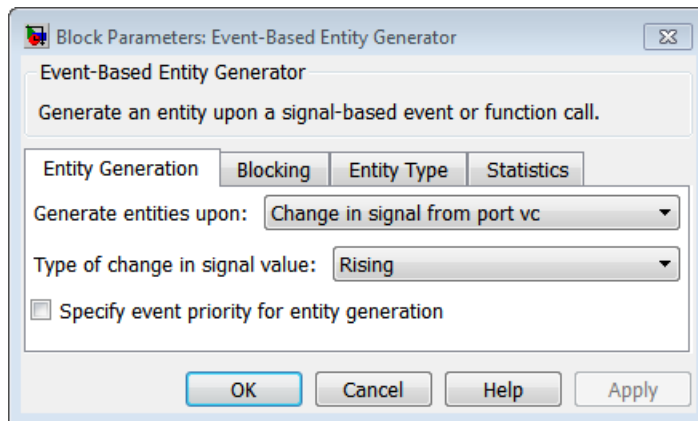
Label	Description
OUT	Port through which generated entities depart.

**Signal Output Ports**

Label	Description	Time of Update When Statistic Is On	Order of Update
#d	.	After entity departure	3
pe	<p>A value of 1 indicates that the block stores an entity that has tried and failed to depart. In that case, the entity is a pending entity.</p> <p>A value of 0 indicates that the block does not store any pending entities.</p>	<p>Sample time hit of 1 occurs after the block stores an entity that has tried and failed to depart.</p> <p>Sample time hit of 0 occurs after the departure or discarding of the pending entity.</p>	1
w	Average intergeneration time, in seconds, for all pairs of successive entities that have departed from this block. The signal value is 0 before the second entity departure.	After entity departure	2

## Dialog Box

### Entity Generation



#### Generate entities upon

The type of event that indicates when the block can generate an entity.

#### Trigger type, Type of change in signal value

**Trigger type** determines whether rising, falling, or either type of trigger edge causes an entity generation. **Generate entities upon** to Trigger from port tr.

**Type of change in signal value** determines whether rising, falling, or either type of value change causes an entity generation. **Generate entities upon** to Change in signal from port vc.

#### Specify event priority for entity generation

Select this option to prioritize the entity-generation event explicitly, relative to other simultaneous events in the simulation.

#### Generation event priority

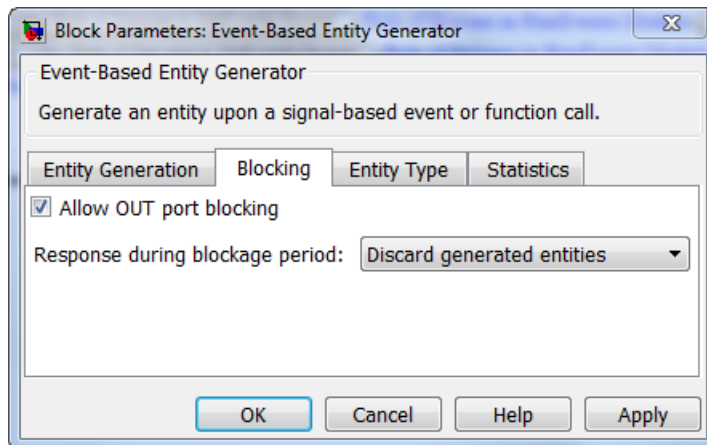
The priority of the entity-generation event, relative to other simultaneous events in the simulation. **Resolve simultaneous signal updates according to event priority.**

#### Allow entity generation upon sample time hit (or function call) at simulation start time

If you select **Allow entity generation upon sample time hit at simulation start time**, the block generates the first entity when the simulation begins. Otherwise, the block generates the first entity upon the first update of the **ts** signal at a nonzero value of time. **Generate entities to Sample time hit** from port **ts**.

If you select **Allow entity generation upon function call at simulation start time**, the block responds to function calls at the starting time of the simulation. Otherwise, the block responds only to function calls at subsequent times. **Generate entities upon** to **Function call** from port **fcn**.

## Blocking



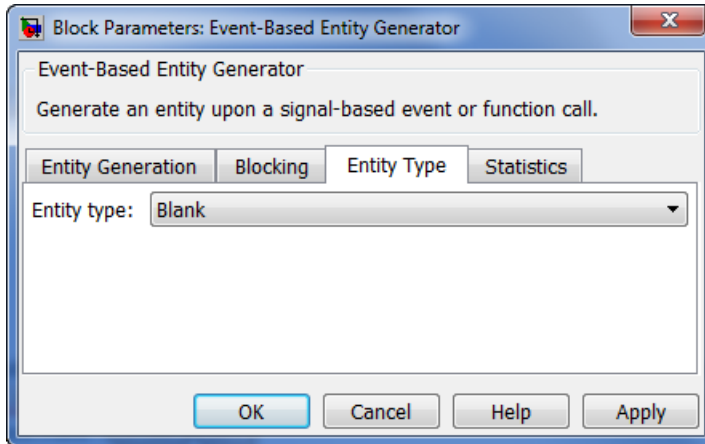
### Allow OUT port blocking

If you do not select this option and a generated entity cannot depart immediately, the simulation halts with an error message.

### Response during blockage period

Determines how the block responds if a generated entity cannot depart immediately; see “Responding to Blockage at the Entity Output Port” on page 2-119. **Allow OUT port blocking**.

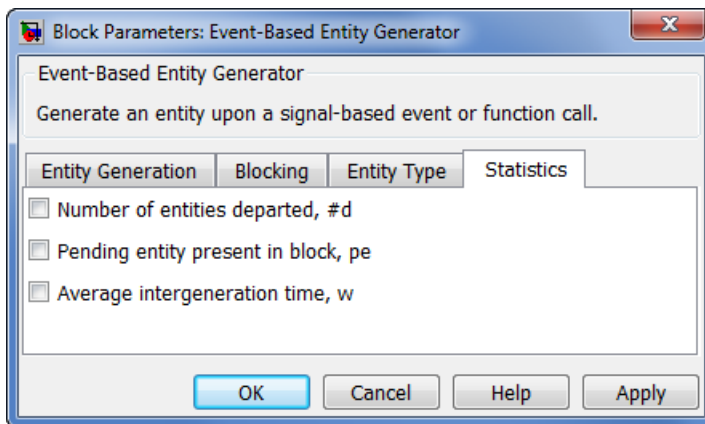
## Entity Type



### Entity type

The blank type includes no attributes. The standard type includes attributes called Priority and Count with default values of 10 and 0, respectively.

## Statistics



### Number of entities departed

**#d.**

**Pending entity present in block**

**pe.**

**Average intergeneration time**

**w.**

## **See Also**

Time-Based Entity Generator (Obsolete)

Entity Sink

“Generate Entities When Events Occur”

**Introduced before R2006a**

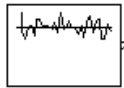
## Event-Based Random Number (Obsolete)

Generate random numbers from specified distribution, parameters, and initial seed

### Library

Generators / Signal Generators

### Description



This block generates random numbers in an event-based manner, inferring from a subsequent block when to generate a new random number. For example, when connected to the **t** input port of a **Single Server** block, the **Event-Based Random Number** block generates a new random number each time an entity arrives at the server.

You specify the distribution from which the block draws random numbers. The seed of the random number generator is reset to the value of the **Initial seed** parameter each time a simulation starts, which makes the random behavior repeatable.

### Connecting to Other Blocks

This block has a restricted set of valid connections to other blocks because the **Event-Based Random Number** block infers from a subsequent block when to generate a new random number.

All indirect connections must be via blocks that have all of the following characteristics:

- Has exactly one input signal
- Has no function-call output signals

---

**Tip** For an indirect connection to the **Atomic Subsystem** block, the restrictions on input and output signals apply to the subsystem itself, not the blocks inside the subsystem.

---



## Distribution Types

The **Distribution** parameter names the type of distribution the block uses to generate random numbers. When you set the **Distribution** parameter, the block changes its dialog box to show additional parameters that determine the probability density function (or probability mass function, for a discrete distribution). The available distributions and the additional parameters for each are described in the sections that follow.

<b>Distribution</b>	<b>Additional Parameters</b>
Exponential	Mean
Uniform	Minimum, Maximum
Bernoulli	Probability of 1
Binomial	Probability of success in a single trial, Number of trials
Triangular	Minimum, Maximum, Mode
Gamma	Threshold, Scale, Shape
Gaussian (normal)	Mean, Standard deviation
Geometric	Probability of success in a single trial
Poisson	Mean
Lognormal	Threshold, Mu, Sigma
Log-logistic	Threshold, Scale
Beta	Minimum, Maximum, Shape parameter a, Shape parameter b
Discrete uniform	Minimum, Maximum, Number of values
Weibull	Threshold, Scale, Shape
Arbitrary continuous	Value vector, Cumulative probability function vector
Arbitrary discrete	Value vector, Probability vector

For information about the definitions and properties of each distribution, see “References” on page 2-135 below.

### Range of Output Values

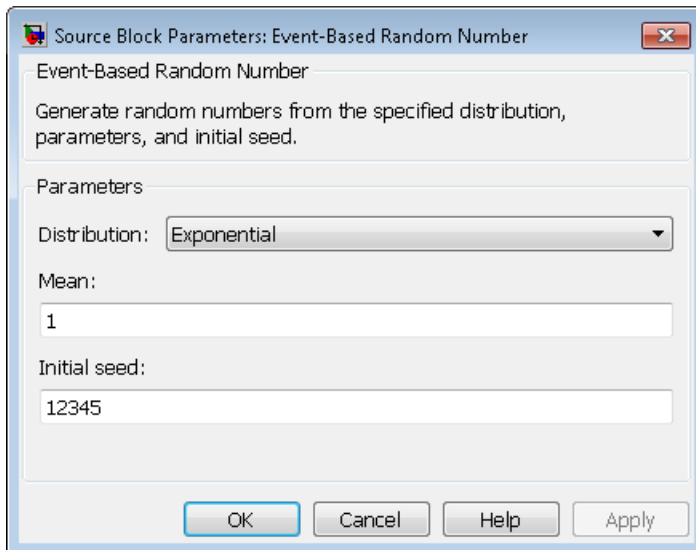
Different distributions have different output ranges. Make sure the distribution and parameters you choose are suitable for your application. For example, when generating random service times, do not use a Gaussian distribution because it can produce negative numbers.

### Ports

This block has one signal output port for the random numbers. The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

The block has no entity ports, and no signal input port.

### Dialog Box



#### Distribution

The distribution from which the block generates random numbers.

#### Mean

The mean value of an exponential, Gaussian, or Poisson distribution.

**Minimum**

The minimum value of a uniform, triangular, beta, or discrete uniform distribution.

**Maximum**

The maximum value of a uniform, triangular, beta, or discrete uniform distribution.

**Probability for output to be 1**

The probability of a one in a Bernoulli distribution.

**Probability of success in a single trial**

The probability of a successful outcome in each trial used to describe a binomial or geometric distribution.

**Number of trials**

The number of trials used to describe a binomial distribution.

**Mode**

The statistical mode of a triangular distribution. The triangular distribution also uses the **Minimum** and **Maximum** parameters to define its density function.

**Threshold, Scale, Shape**

Parameters that define the density function of a gamma, log-logistic, or Weibull distribution. The log-logistic distribution does not use a **Shape** parameter, however.

**Threshold, Mu, Sigma**

Parameters that define the density function of a lognormal distribution. The log of a lognormal random variable is normally distributed with mean **Mu** and standard deviation **Sigma**.

**Standard deviation**

The standard deviation of a Gaussian distribution, which also uses the **Mean** parameter to define its density function.

**Shape parameter a, Shape parameter b**

The first and second shape parameters, respectively, of a beta distribution. The beta distribution also uses the **Minimum** and **Maximum** parameters to define its density function.

**Number of values**

The number of possible outputs of a discrete uniform distribution, including the values of the **Minimum** and **Maximum** parameters. **Number of values** must exceed 1.

**Value vector**

A vector of values in ascending order, representing the possible random values in an arbitrary continuous or arbitrary discrete distribution.

**Cumulative probability function vector**

A vector of values in ascending order representing the cumulative probability function for an arbitrary continuous distribution. The first and last values of the vector must be 0 and 1, respectively. This parameter and the **Value vector** parameter must have the same vector length.

**Probability vector**

A vector of values representing the probability of each value in the **Value vector** function for an arbitrary discrete distribution. This vector must contain nonnegative values that sum to 1. This parameter and the **Value vector** parameter must have the same vector length.

**Initial seed**

A nonnegative integer that initializes the random number generator.

## Algorithm

Below are the expressions for  $f$ , the probability density functions for the continuous distributions and probability mass functions for the discrete distributions that the block supports.

**Exponential Distribution**

$$f(x) = \begin{cases} \frac{1}{\mu} \exp\left(-\frac{x}{\mu}\right) & \text{for } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where  $\mu$  is the **Mean** parameter, a positive number.

A similar function in the Statistics and Machine Learning Toolbox™ software is `expnd`.

**Uniform Distribution**

$$f(x) = \begin{cases} \frac{1}{U-L} & \text{for } L \leq x \leq U \\ 0 & \text{otherwise} \end{cases}$$

where  $L$  is the **Minimum** parameter and  $U$  is the **Maximum** parameter.

Similar functions are `rand` in MATLAB software and `unifrnd` in the Statistics and Machine Learning Toolbox software.

## Bernoulli Distribution

$$f(x) = \begin{cases} p^x(1-p)^{1-x} & \text{for } x = 0, 1 \\ 0 & \text{otherwise} \end{cases}$$

where  $p$  is the **Probability of 1** parameter. The value  $p$  must be between 0 and 1, inclusive. This is a discrete distribution.

This distribution is a special case of the binomial distribution in which the number of trials is 1.

## Binomial Distribution

$$f(x) = \begin{cases} \frac{n!}{x!(n-x)!} p^x q^{(n-x)} & \text{for } x = 0, 1, 2, \dots, n \\ 0 & \text{otherwise} \end{cases}$$

where  $p$  is the **Probability of success in a single trial** parameter,  $q = 1-p$ , and  $n$  is the **Number of trials** parameter. The value  $p$  must be between 0 and 1, inclusive, while  $n$  must be positive. This is a discrete distribution.

A similar function in the Statistics and Machine Learning Toolbox software is `binornd`.

## Triangular Distribution

$$f(x) = \begin{cases} \frac{2(x-L)}{(U-L)(m-L)} & \text{for } L \leq x \leq m \\ \frac{2(U-x)}{(U-L)(U-m)} & \text{for } m < x \leq U \\ 0 & \text{otherwise} \end{cases}$$

where  $L$  is the **Minimum** parameter,  $U$  is the **Maximum** parameter, and  $m$  is the **Mode** parameter. These parameters must satisfy  $L < m < U$ .

## Gamma Distribution

$$f(x) = \begin{cases} \frac{\left(\frac{x-\theta}{b}\right)^{a-1} \exp\left(-\frac{x-\theta}{b}\right)}{b\Gamma(a)} & \text{for } x \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where  $\theta$  is the **Threshold** parameter,  $b$  is the **Scale** parameter, and  $a$  is the **Shape** parameter. The **Scale** and **Shape** parameters must be positive. Also,  $\Gamma$  is the gamma function (**gamma** in MATLAB code).

A similar function in the Statistics and Machine Learning Toolbox software is `gamrnd`.

## Gaussian (Normal) Distribution

$$f(x) = \frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{\sigma\sqrt{2\pi}}$$

where  $\mu$  is the **Mean** parameter and  $\sigma$  is the **Standard deviation** parameter. The standard deviation parameter must be nonnegative.

Similar functions are `randn` in MATLAB software and `normrnd` in the Statistics and Machine Learning Toolbox software.

## Geometric Distribution

If the **Probability of success in a single trial** parameter is strictly between 0 and 1, then the probability mass function is defined by

$$f(x) = \begin{cases} pq^x & \text{for } x = 0, 1, 2, \dots \\ 0 & \text{otherwise} \end{cases}$$

where  $p$  is the **Probability of success in a single trial** parameter and  $q = 1-p$ .

In the special case where the **Probability of success in a single trial** parameter is 1, then

$$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ 0 & \text{otherwise} \end{cases}$$

This is a discrete distribution.

A similar function in the Statistics and Machine Learning Toolbox software is `geornd`.

## Poisson Distribution

$$f(x) = \begin{cases} \frac{e^{-\lambda} \lambda^x}{x!} & \text{for } x = 0, 1, 2, \dots \\ 0 & \text{otherwise} \end{cases}$$

where  $\lambda$  is the **Mean** parameter, a positive number. This is a discrete distribution.

A similar function in the Statistics and Machine Learning Toolbox software is `poissrnd`.

## Lognormal Distribution

$$f(x) = \begin{cases} \frac{\exp\left[-\frac{(\ln(x-\theta)-\mu)^2}{2\sigma^2}\right]}{(x-\theta)\sigma\sqrt{2\pi}} & \text{for } x \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where  $\theta$  is the **Threshold** parameter,  $\mu$  is the **Mu** parameter, and  $\sigma$  is the **Sigma** parameter. The **Sigma** parameter must be positive.

A similar function in the Statistics and Machine Learning Toolbox software is `lognrnd`.

## Log-Logistic Distribution

The log-logistic distribution is derived from the logistic distribution, as follows:  
 $X$  = Random variable with logistic distribution

$Y = e^X$  = Random variable with log-logistic distribution

The probability density function for the logistic distribution is

$$f_{\text{logistic}}(x) = \frac{1}{b} \cdot \frac{e^{(x-\theta)/b}}{\left(1 + e^{(x-\theta)/b}\right)^2}$$

where  $\theta$  is the **Threshold** parameter and  $b$  is the **Scale** parameter. The **Scale** parameter must be positive.

### Beta Distribution

$$f(x) = \begin{cases} \frac{(x-L)^{a-1}(U-x)^{b-1}}{B(a,b)(U-L)^{a+b-1}} & \text{for } L \leq x \leq U \\ 0 & \text{otherwise} \end{cases}$$

where  $L$  is the **Minimum** parameter,  $M$  is the **Maximum** parameter,  $a$  is the **Shape parameter a** parameter,  $b$  is the **Shape parameter b** parameter, and  $B(a,b)$  is the beta function defined by

$$B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1} dt$$

The two shape parameters must be positive.

A similar function in the Statistics and Machine Learning Toolbox software is `betarnd`.

### Discrete Uniform Distribution

$$f(x) = \begin{cases} 1 / K & \text{for } x = L + k \frac{(U-L)}{K-1}, k = 0, 1, 2, \dots, K-1 \\ 0 & \text{otherwise} \end{cases}$$

where  $L$  is the **Minimum** parameter,  $U$  is the **Maximum** parameter, and  $K$  is the **Number of values** parameter. This is a discrete distribution. If  $(U-L)/(K-1)$  and  $L$  are both integers, then all outputs from this distribution are integers.



Similar functions are `randi` in MATLAB software and `unidrnd` in the Statistics and Machine Learning Toolbox software.

## Weibull Distribution

$$f(x) = \begin{cases} \frac{\gamma}{\alpha} \left( \frac{x-\theta}{\alpha} \right)^{\gamma-1} \exp \left[ - \left( \frac{x-\theta}{\alpha} \right)^\gamma \right] & \text{for } x \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where  $\theta$  is the **Threshold** parameter,  $\alpha$  is the **Scale** parameter, and  $\gamma$  is the **Shape** parameter. The **Scale** and **Shape** parameters must be positive.

A similar function in the Statistics and Machine Learning Toolbox software is `wblrnd`.

## References

- [1] Evans, M., N. Hastings, and B. Peacock. *Statistical Distributions*. Wiley-Interscience, 2000.
- [2] Johnson, N. L., S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, Volume 1. Wiley-Interscience, 1993.
- [3] Johnson, N. L., S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, Volume 2. Wiley-Interscience, 1994.
- [4] Johnson, N. L., S. Kotz, and A. W. Kemp. *Univariate Discrete Distributions*. Wiley-Interscience, 1993.

## See Also

Signal Latch (Obsolete), Event-Based Sequence (Obsolete)

Introduced before R2006a

## Event-Based Sequence (Obsolete)

Generate sequence of numbers from specified column vector

### Library

Generators / Signal Generators

### Description

This block generates an event-based signal using data you provide, inferring from a subsequent block when to output the next value from your data. You specify the data as a column vector using the **Vector of output values** parameter. The parameter value can be any MATLAB language expression that evaluates to a column vector, including the name of a column vector variable in the MATLAB base workspace. As an example of inferring timing from a subsequent block, if you connect this block to the **t** input port of a **Single Server** block, then the **Event-Based Sequence** block outputs a new value each time an entity arrives at the server.

### Behavior After Data Runs Out

If the block needs more data than the vector contains, subsequent output values follow a rule you specify using the **Form output after final data value by** parameter. The table below lists possible values for this parameter.

---

**Note:** In all cases, the choice of parameter value affects only the values, not the timing, of the output signal. The output signal is always an event-based signal whose sample time hits depend on notifications from a subsequent block.

---

Parameter Value	Description
Cyclic repetition	When the block needs a new output value after exhausting the data, it starts over at the beginning of the vector.

Parameter Value	Description
Holding final value	After exhausting the data, the block outputs the last data value for every sample time hit.
Setting to infinity	After exhausting the data, the block outputs the value <code>inf</code> for every sample time hit. For example, if the block outputs intergeneration times for an entity generator, then the generator produces up to a fixed number of entities.
Setting to zero	After exhausting the data, the block outputs zero for every sample time hit. For example, if the block outputs service times for a server, then the server delays up to a fixed number of entities.

## Connecting to Other Blocks

This block has a restricted set of valid connections to other blocks because the `Event-Based Sequence` block infers from a subsequent block when to generate a new random number.

All indirect connections must be via blocks that have all of the following characteristics:

- Has exactly one input signal
- Has no function-call output signals

---

**Tip** For an indirect connection to the `Atomic Subsystem` block, the restrictions on input and output signals apply to the subsystem itself, not the blocks inside the subsystem.

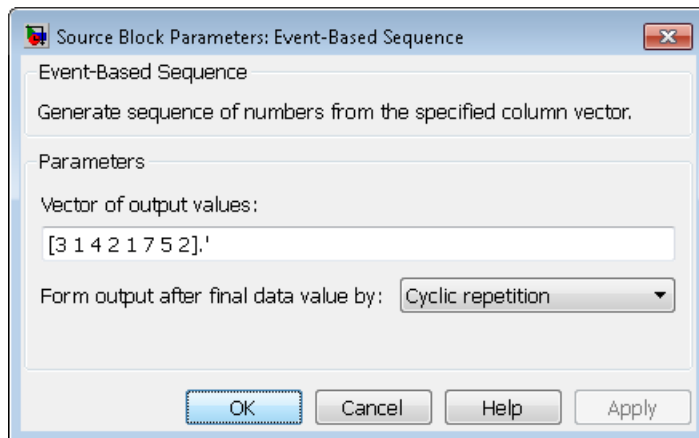
---

## Ports

This block has one signal output port for the numbers in the sequence. The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

The block has no entity ports, and no signal input port.

### Dialog Box



#### Vector of output values

A column vector whose entries become values of this block's output signal. To use a column vector variable in the MATLAB base workspace, enter the variable name.

#### Form output after final data value by

The method for generating output after the block exhausts the data referenced in the **Vector of output values** parameter.

### Examples

- “Set Attributes”

### See Also

Event-Based Random Number (Obsolete), Repeating Sequence Stair, From Workspace

**Introduced in R2006b**

# Event Filter (Obsolete)

Conditionalize, suppress, or prioritize execution of Atomic Subsystem

## Library

SimEvents Ports and Subsystems



## Description Event Filter

This block influences an **Atomic Subsystem** block by specifying the signal-based events upon which the subsystem executes. This block can also prioritize the execution of a subsystem with regard to other events occurring simultaneously by scheduling a subsystem execution on the event calendar. Consider an event-based signal that is an input to an **Atomic Subsystem** block. Without the **Event Filter** block, every sample time hit of the signal causes the subsystem to execute immediately. Inserting the **Event Filter** block on that signal line enables you to influence the subsystem behavior as follows:

- Specify the type of signal-based event that causes the subsystem to execute. Choices are:
  - Sample time hit
  - Change in signal value (rising, falling, or either)
  - Trigger (rising, falling, or either)

If the input signal of this block is a nonscalar array, the block detects one qualifying event if any of the positions in the array has a qualifying event. For example, a change in signal value from [ 1 2 3 ] to [ 1 5 6 ] represents one qualifying event, not two. If N distinct qualifying events occur at distinct sample time hits in the input signal of this block, the subsystem executes N times and updates its output signals N times.

- Prevent the input signal of this block from causing the subsystem to execute. In this case, the signal passively provides data to the subsystem. The subsystem can still execute based on signal-based events of a different input signal.

- Prioritize the subsystem execution, relative to other simultaneous events in the simulation. Instead of occurring immediately upon a signal-based event, the execution becomes a scheduled event on the event calendar.

### Connecting to Other Blocks

The output port of this block can connect to only one input port of an `Atomic Subsystem` block. The connection line cannot branch.

### Behavior During Simulation

When the input signal of an `Event Filter` block has a sample time hit, it does the following:

- 1 Updates its output signal with the value of the input signal. This value is available to the `Atomic Subsystem` block to which the `Event Filter` block connects.
- 2 Determines whether to execute the `Atomic Subsystem` block, based on the settings in the block dialog box of the `Event Filter` block. If the `Event Filter` block is not supposed to execute the `Atomic Subsystem` block, the `Event Filter` does nothing further, until the next sample time hit of the input signal. Otherwise, processing continues to the next step.
- 3 Determines when to execute the `Atomic Subsystem` block.
  - If you did not select the **Resolve simultaneous signal updates according to event priority** option, the `Event Filter` block executes the `Atomic Subsystem` block immediately.
  - If you select the **Resolve simultaneous signal updates according to event priority** option, the `Event Filter` block schedules an event on the event calendar. The event time is the current simulation time. The event priority is the value of the **Event priority** parameter in the `Event Filter` block. When the event calendar executes this event, the `Atomic Subsystem` block performs its computation.
  - If you select both the **Resolve simultaneous signal updates according to event priority** option, and the configuration parameter **Prevent duplicate events on multiport blocks and branched signals**, the software uses the **Event priority** parameter to help Simulink to sort blocks in the model. In this case, the software no longer schedules an event on the event calendar.

## Ports

### Signal Input Ports

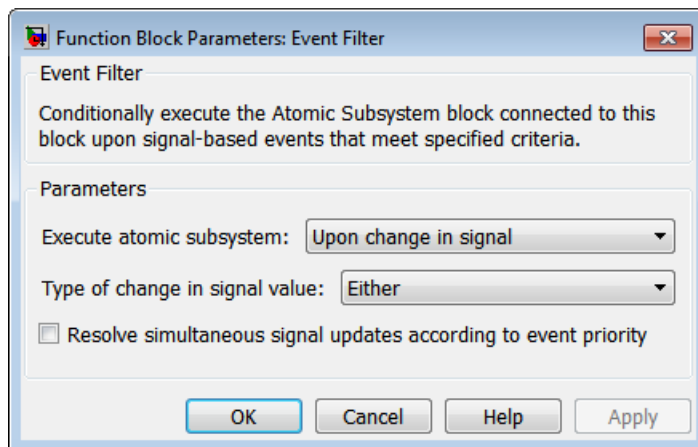
Label	Description
None	Event-based signal. The signal can have any fixed dimension or complexity. It has data type double.

### Signal Output Ports

Label	Description
None	Event-based signal whose value matches that of the input signal. This output signal connects to an input port of an <b>Atomic Subsystem</b> block. When the <b>Event Filter</b> block detects a qualifying signal-based event in its input signal, the subsystem executes immediately or the block schedules an execution event on the event calendar. (Block parameters determine the type of event that qualifies and the choice of immediate or scheduled execution.)

The initial output value is the same as that of the input signal. This value is in effect before the first sample time hit of the input signal.

## Dialog Box



### Execute atomic subsystem

Determines what constitutes a qualifying event in the input signal of this block. If the signal is complex, you must select `Upon sample time hit` or `Never`.

### **Trigger type**

The type of trigger that further restricts the event type specified in **Execute atomic subsystem**. You see this field only if you set **Execute atomic subsystem** to `Upon trigger`.

### **Type of change in signal value**

The type of change in the signal value that further restricts the event type specified in **Execute atomic subsystem**. You see this field only if you set **Execute atomic subsystem** to `Upon change in signal`.

### **Resolve simultaneous signal updates according to event priority**

Select this option to control the sequencing of the subsystem execution in response to updates in the input signal of this block, relative to other simultaneous events in the simulation. If you do not select this option, the application executes the subsystem immediately upon detecting the signal-based event.

### **Event priority**

The priority of the subsystem execution event (in response to updates in the input signal of this block), relative to other simultaneous events in the simulation.

Use of this parameter depends on the following:

- You see this field only if you select **Resolve simultaneous signal updates according to event priority**.
- If you also select the configuration parameter **Prevent duplicate events on multiport blocks and branched signals** in your model, the software uses the **Event priority** parameter to help Simulink to sort blocks in the model. In this case, the software does not schedule an event that you can view on the SimEvents event calendar.

### **Introduced in R2011b**



# Event to Timed Signal (Obsolete)

Convert event-based signal to time-based signal

## Library

Gateways

## Description

This block converts an event-based data signal into a time-based data signal. The output signal assumes exactly one value at any given time on the simulation clock. The output signal is almost identical to the input signal, except:

- The output signal omits zero-duration values, if any, from the input signal.
- The output signal has a sample time type of “fixed in minor step.” As a result, the output signal might have sample time hits at times unrelated to the input signal but related to other time-based signals in the model.
- The output signal is suitable for modeling time-based dynamics. The signal cannot be an input to a block that requires an event-based input signal. Blocks that can process either time-based or event-based signals might process them differently.
- The initial output value is the same as the initial input value. However, if the input signal is undefined at  $T = 0$ , as in the case of an atomic subsystem that has an event-based input signal, the output signal of this block has an initial output of 0.

## Ports

### Signal Input Ports

Label	Description
None	Event-based signal. The signal can have any fixed dimension, complexity, or data type.

### Signal Output Ports

Label	Description
None	Time-based signal

## See Also

Timed to Event Signal (Obsolete)

“Time-Based Signals and SimEvents Block Transitions”

**Introduced in R2011b**

# Event to Timed Function-Call (Obsolete)

Convert event-based function call to time-based function call

## Library

Gateways

## Description

This block converts a scalar event-based function call into a time-based function call. The output signal is almost identical to the input signal, except that the output can be an input to a block that requires a time-based function-call input signal.

## Ports

### Signal Input Ports

Label	Description
None	Event-based function-call signal.

### Signal Output Ports

Label	Description
None	Time-based function-call signal.

## See Also

Timed to Event Function-Call (Obsolete)

“Time-Based Signals and SimEvents Block Transitions”

Introduced in R2011b

## FIFO Queue (Obsolete)

Store entities in sequence for undetermined length of time

### Library

Queues



This block stores up to  $N$  entities simultaneously, where  $N$  is the **Capacity** parameter value. The block attempts to output an entity through the **OUT** port, but retains the entity if the **OUT** port is blocked. If the block is storing multiple entities and no entity times out, then entities depart in a first-in, first-out (FIFO) fashion. If an entity in this block is scheduled to time out, then it might depart prematurely via the optional **TO** port. The length of time that an entity stays in this block cannot be determined in advance.

The **IN** port is unavailable whenever this block stores exactly  $N$  entities. In this case, the queue is said to be full.

### Ports

#### Entity Input Ports

Label	Description
IN	Port for arriving entities, which are stored.

#### Entity Output Ports

Label	Description
OUT	Port for departing entities that do not time out while in this block.

Label	Description
<b>TO</b>	Port for entities that time out while in this block. You see this port only if you select <b>Enable TO port for timed-out entities</b> . This port must not be blocked when an entity attempts to depart here.

### Signal Output Ports

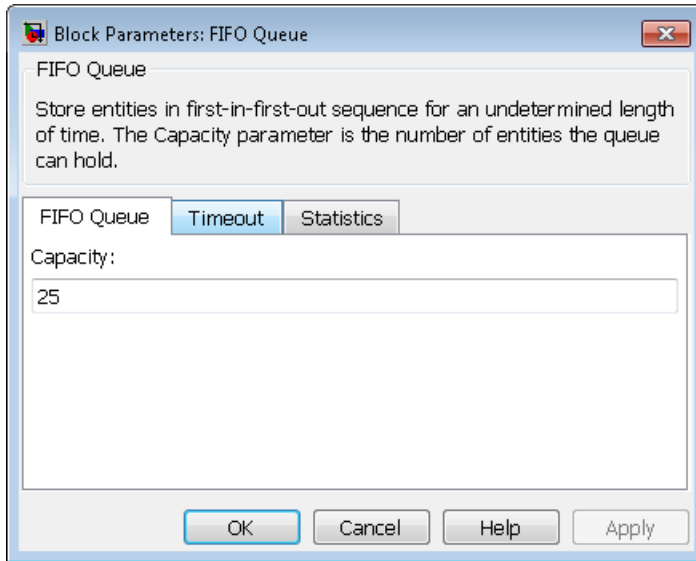
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
<b>#d</b>	Number of entities that have departed from this block via the <b>OUT</b> port since the start of the simulation.	After entity departure via the <b>OUT</b> port	3
<b>#n</b>	Number of entities currently in the queue.	After entity arrival and after entity departure	2
<b>w</b>	Sample mean of the waiting times in this block for all entities that have departed via any port.	After entity departure	1
<b>len</b>	Average number of entities in the queue over time, that is, the time average of the <b>#n</b> signal.	After entity arrival and after entity departure.	1
<b>#to</b>	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the <b>TO</b> port	3

Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

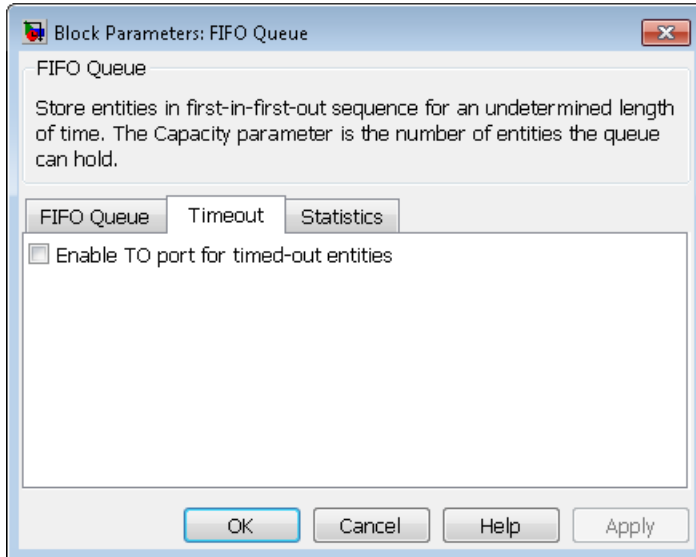
### FIFO Queue Tab



### Capacity

Determines how many entities the block can store at a time. The capacity must be a positive integer or Inf.

## Timeout Tab

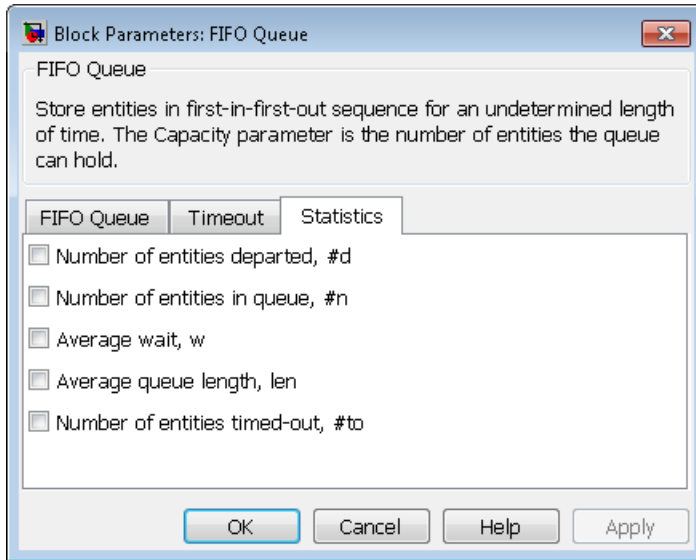


### Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the **Schedule Timeout (Obsolete)** block.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see the preceding table, “Signal Output Ports”.



### **Number of entities departed**

Allows you to use the signal output port labeled **#d**.

### **Number of entities in queue**

Allows you to use the signal output port labeled **#n**.

### **Average wait**

Allows you to use the signal output port labeled **w**.

### **Average queue length**

Allows you to use the signal output port labeled **len**.

### **Number of entities timed out**

Allows you to use the signal output port labeled **#to**.

## **Examples**

- “Build a Discrete-Event Model”
- “Constructs Involving Queues and Servers”



## See Also

LIFO Queue (Obsolete), Priority Queue (Obsolete)  
“Storage”

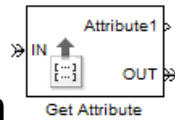
**Introduced before R2006a**

## Get Attribute (Obsolete)

Output value of entity attribute

### Library

Attributes

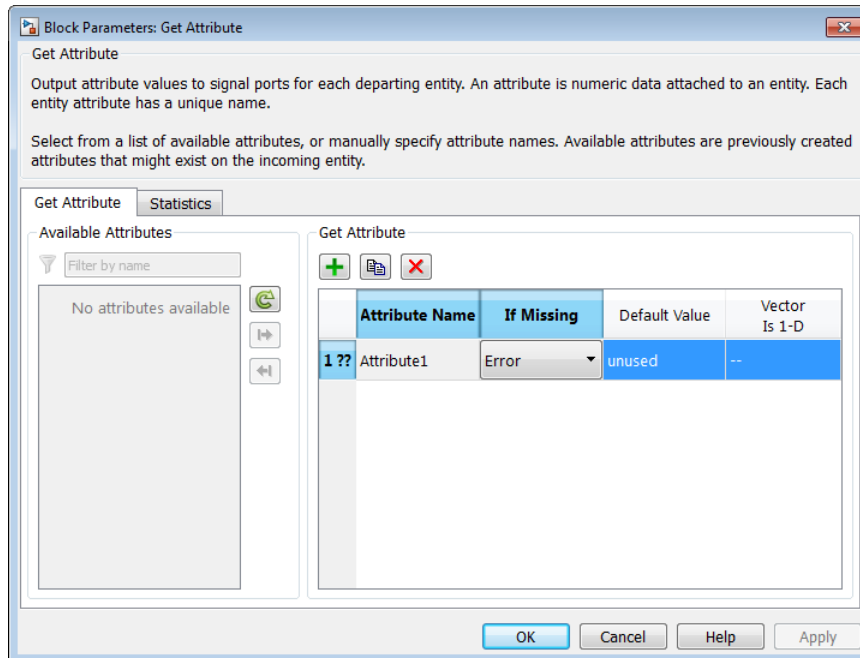


### Description

This block outputs signals using data from entity attributes. For each arriving entity, the block updates the signal at the signal output ports using values from attributes named in the block dialog box. The block also outputs the entity unchanged.

# Dialog Box

## Get Attribute Tab



### Available Attributes




Use the **Available Attributes** controls to:

- Select the attributes from incoming entity paths that you want to access.
- Add the attributes to the **Get Attribute** table, where you can modify them.

The list displays all the attributes on all the incoming entities. (If the entity paths entering the Get Attribute block do not have any attributes, the **Available Attributes** list is empty).

If the attribute list is long, you can type the attribute name in the text box to filter the list.

Use the buttons in the **Available Attributes** section to help build the attributes table. The buttons perform these actions:

Button	Action
	Refresh the <b>Available Attributes</b> list. This action updates the list with any upstream model changes you make while the block dialog box is open.
	Add the selected attribute to the <b>Get Attribute</b> table.
	Move the selected attribute from the <b>Get Attribute</b> table to the <b>Available Attributes</b> list.  Note: If the selected attribute is one you added manually, this button appears dimmed.

The message area below the available attributes list displays additional messages about the attributes, as they apply.

Message	Meaning
> Attribute already selected	You have already added the attribute to the <b>Get Attribute</b> table. You cannot add the attribute to the table again.
* Attribute may not be present	When multiple entity paths enter the block, all entities might not have the same attributes. Attributes that are not on all entering entities display an asterisk in the list, and this message appears. If you add such an attribute to the <b>Get Attribute</b> table, the behavior depends on how the <b>If Missing</b> field is set.

### Get Attribute




Use the controls under **Get Attribute** to build and manage the list of attributes to access on each incoming entity. Each attribute appears as a row in a table.

Using these controls, you can:

- Specify an attribute manually to access on the entity.

- Modify an attribute that you added to the table from the **Available Attributes** list to access on the entity.

The buttons under **Get Attribute** perform these actions:

Button	Action	Notes
	Add a template attribute to the table.	Rename the attribute and specify its properties.
	Add a copy of the selected attribute to the table to use as the basis of a new attribute.	Rename the copy. Two attributes cannot have the same name.
	Remove the selected attribute from the <b>Get Attribute</b> table.	When you delete an attribute this way, no confirmation appears and you cannot undo the operation.

**Note:** If you delete a row and apply the change, the deletion can affect signal output ports corresponding to other attributes. For example, if the block has a signal output port **A2** and you delete the attribute with a port marked **A1**, the block relabels **A2** as **A1**. Verify that any signal that connects to the relabeled port is still connected as you expect.

The table displays the attributes you added from the **Available Attributes** list or added manually. Use it to set these four attribute properties:

Property	Specify	Use
<b>Attribute Name</b>	<p>The name of the attribute to access. Each attribute must have a unique name.</p> <p>If the attribute name does not match an attribute listed in <b>Available Attributes</b>, the block displays <b>??</b> next to the attribute name. This symbol denotes that the attribute is not present on incoming</p>	Double-click the existing name, and then type the new name.

Property	Specify	Use
	entity paths. The simulation behavior you see depends on the value of <b>If Missing</b> (see “Missing Attributes” on page 2-156).	
<b>If Missing</b>	The response of the block when the entity does not have an attribute named in the table.	Select a block response from the list.
<b>Default Value</b>	The value for the corresponding output signal if the entity does not have an attribute specified in the table. To learn about the kind of data you can use as a default value, see “Attribute Value Support”. You can set this field only if you set <b>If Missing</b> to <b>Default value</b> or <b>Warn</b> .	Double-click the field and type a value.
<b>Vector Is 1-D</b>	Whether the block considers the default value as a vector of length N when <b>Default Value</b> evaluates to an N-element row or column vector. Otherwise, the block considers the default value as a multidimensional array. This option affects attributes whose <b>If Missing</b> parameter is set to <b>Default value</b> or <b>Warn</b> .	Select the check box to treat the attribute as a vector of length N. Clear it to treat the attribute as a multidimensional array.

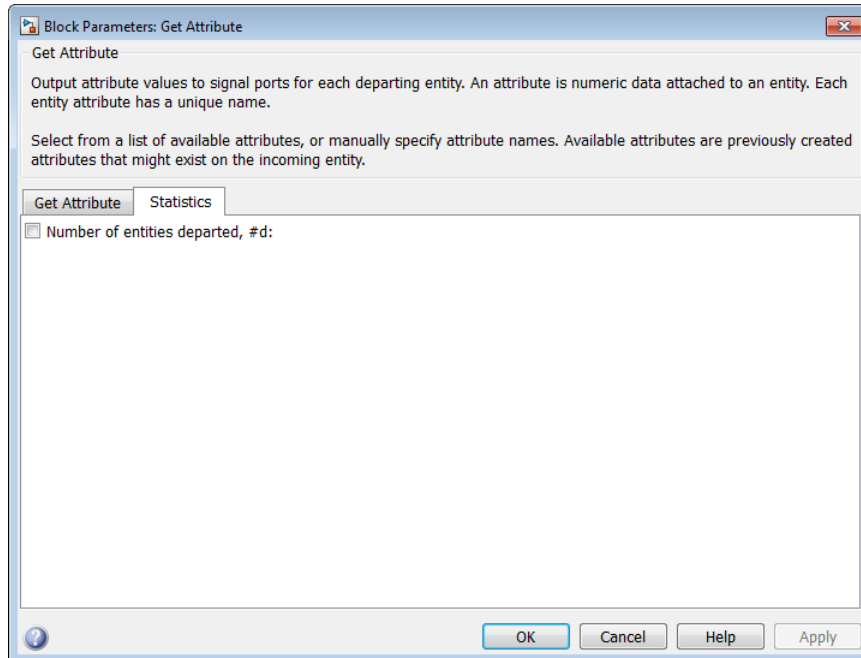
## Missing Attributes

You can specify the block behavior if the arriving entity does not have an attribute listed in the table of the block dialog box. Use the **If Missing** parameter for that attribute.

Parameter Value	Block Behavior in Case of Missing Attribute
Error	The block issues an error message and halts simulation. In this case, the <b>Default Value</b> and <b>Treat Vector as 1-D</b> parameters are disabled.
Default value	The block outputs a default value that you specify using the <b>Default Value</b> and <b>Treat Vector as 1-D</b> parameters. The simulation proceeds.
Warn	The block outputs a default value that you specify using the <b>Default Value</b> and <b>Treat Vector as 1-D</b> parameters. The block also issues a warning in the MATLAB Command Window. The simulation proceeds.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see Signal Output Ports.



### Number of entities departed

Allows you to use the signal output port labeled **#d**.

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities

### Entity Output Ports

Label	Description
OUT	Port for departing entities

### Signal Output Ports

Label	Description	Time of Update When Statistic Is On
<b>#d</b>	Number of entities that have departed from this block since the start of the simulation.	After entity departure
<i>Attribute name</i>	Value of the attribute of the same name specified in the table. The default <b>Name</b> that corresponds to each row is <b>Attribute<math>x</math></b> , where $x = 1, 2, 3$ , etc.	After entity departure

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## See Also

Set Attribute (Obsolete)

“Manipulate Entity Attributes”

**Introduced before R2006a**



# Infinite Server (Obsolete)

Delay any number of entities for period of time

## Library

Servers

## Description



This block serves any number of entities for a period of time, called the *service time*, and then attempts to output them through the **OUT** port. If the **OUT** port is blocked, then the block holds the entities until the port becomes unblocked. If an entity in this block is scheduled to time out, then it might depart prematurely via the optional **TO** port.

An infinite server is like an infinite set of single servers connected in parallel, followed by a path combiner; the path combiner notifies entities of an unblocked path in the sequence in which the entities completed their service time, until one entity departs.

You specify the service time, which is the duration of service, via a parameter, attribute, or signal, depending on the **Service time from** parameter value. The block determines the service time for an entity upon its arrival. Service times are assumed to be specified in seconds.

---

**Note:** If you specify the service time via an event-based signal, be sure that its updates occur before the entity arrives.

---

The **IN** port of an infinite server is always available. You can interpret an infinite server as a mechanism for delaying entities. Some discussions of this block suggest this interpretation by using the word *delay* instead of *serve*.

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities, which will be served.

### Signal Input Ports

Label	Description
t	Service time, in seconds, for a newly arrived entity. This signal must be an event-based signal. You see this port only if you set <b>Service time from</b> to <b>Signal port t</b> .

### Entity Output Ports

Label	Description
OUT	Port for departing entities that have completed their service time and have not timed out while in this block.
TO	Port for entities that time out while in this block. You see this port only if you select <b>Enable TO port for timed-out entities</b> . This port must not be blocked when an entity attempts to depart here.

### Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block via the <b>OUT</b> port since the start of the simulation.	After entity departure via the <b>OUT</b> port	5
#n	Number of entities in the block.	After entity arrival and after entity departure	4
pe	A value of 1 indicates that the block stores at least one entity that has tried and failed to depart. Such entities are pending entities.	After the block stores an entity that has tried and failed to depart. In this case, the signal value is 1.  After the departure of a pending entity. In this case, the signal value depends on whether any	1

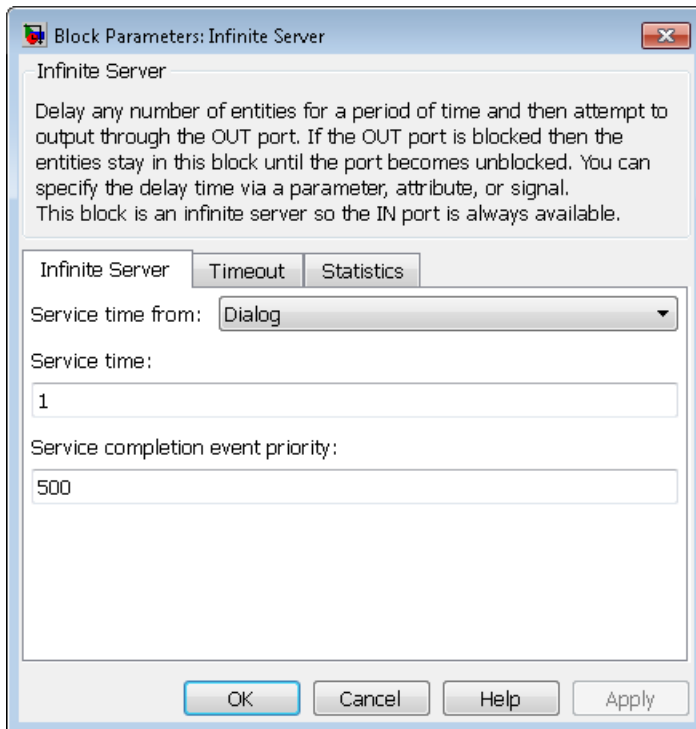
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
	A value of 0 indicates that the block does not store any pending entities.	other pending entities remain in the block.	
<b>#pe</b>	Number of pending entities in the block.	After the block stores an entity that has tried and failed to depart.  After the departure of a pending entity.	3
<b>w</b>	Sample mean of the waiting times in this block for all entities that have departed via any port. An entity's waiting time might exceed its service time if the <b>OUT</b> port is blocked when the entity completes service.	After entity departure	2
<b>#to</b>	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the <b>TO</b> port	5

Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

### Infinite Server Tab



#### Service time from

Determines whether the service time is computed from a parameter in this dialog box, a signal input port, or an attribute of the entity being served.

#### Service time

The service time, in seconds, for all entities. You see this field only if you set **Service time from** to **Dialog**.

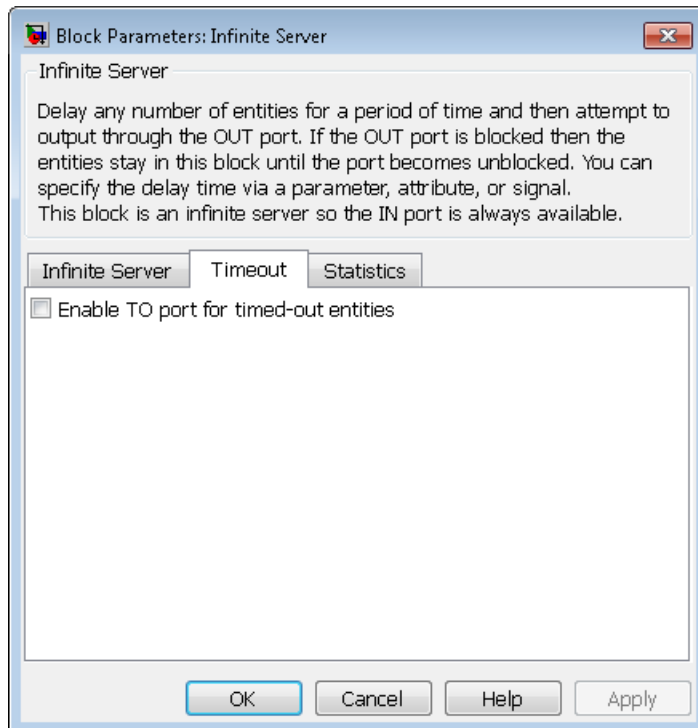
#### Attribute name

The name of the attribute whose value the block uses as the service time for an entity. You see this field only if you set **Service time from** to **Attribute**.

#### Service completion event priority

The priority of the service completion event, relative to other simultaneous events in the simulation.

## Timeout Tab

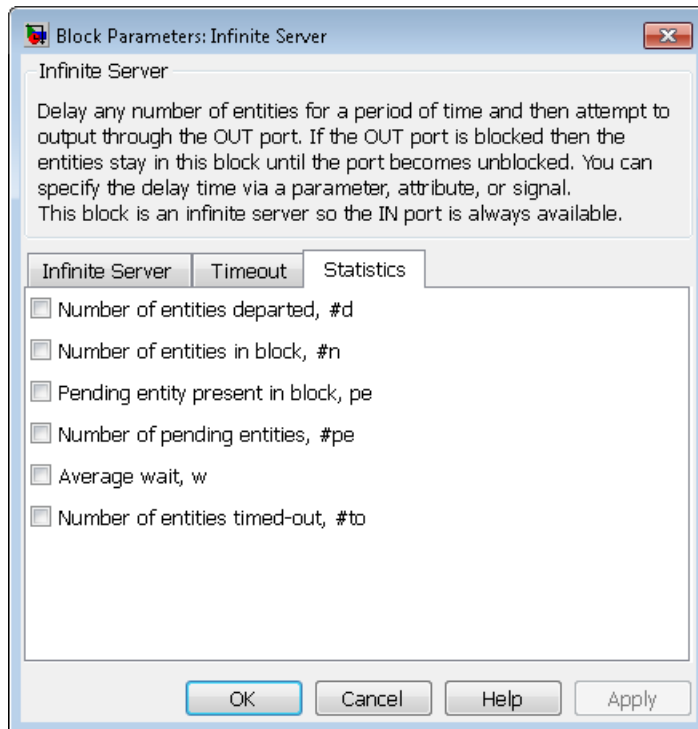


### Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the **Schedule Timeout (Obsolete)** block.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports.



### **Number of entities departed**

Allows you to use the signal output port labeled **#d**.

### **Number of entities in block**

Allows you to use the signal output port labeled **#n**.

### **Pending entity present in block**

Allows you to use the signal output port labeled **pe**.

### **Number of pending entities**

Allows you to use the signal output port labeled **#pe**.

### **Average wait**

Allows you to use the signal output port labeled **w**.

### **Number of entities timed out**

Allows you to use the signal output port labeled **#to**.

## See Also

Single Server (Obsolete), N-Server (Obsolete)

“Write Events Actions”

**Introduced before R2006a**

## Initial Value (Obsolete)

Output specified value until first sample time hit

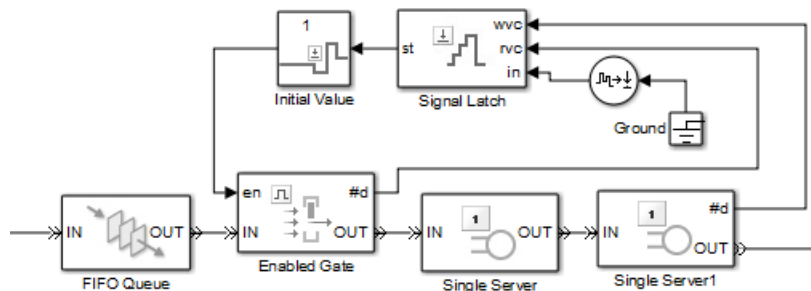
### Library

Signal Management



This block establishes an initial value for an event-based signal. Before the first sample time hit at the input port, the value of the output signal is the **Value until first sample time hit** parameter value. Starting from the first sample time hit, the output signal is identical to the input signal.

The following model fragment illustrates block usage in a feedback loop. When the simulation starts, the **Initial Value** block provides an initial value of 1 that opens the gate to permit the first entity to advance into the feedback loop. Without a nonzero initial value, no entity would arrive at the servers and the **Signal Latch** block would never experience any events.




---

**Note:** The IC block in the Simulink library set operates in a time-based manner and is not suitable for event-based signals.

---



## Ports

### Signal Input Ports

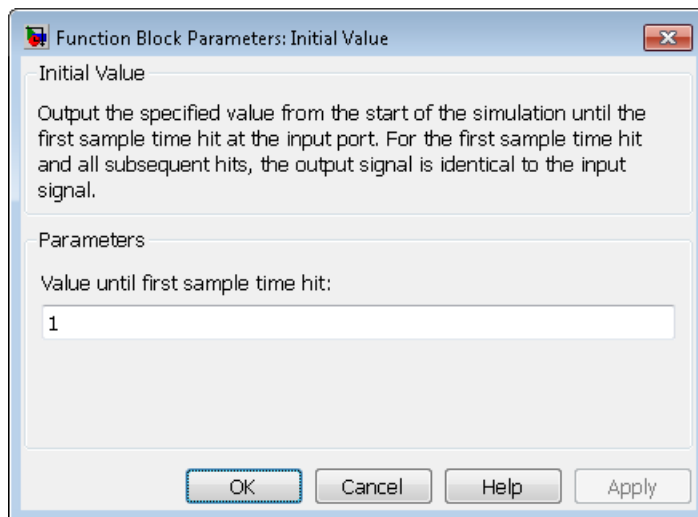
Label	Description
None	The first sample time hit in this signal causes the block to stop using the initial value from the block dialog box. From then on, the output signal is identical to the input signal. This signal must be an event-based signal.

### Signal Output Ports

Label	Description
None	The value is either the initial value in the block dialog box or the input signal value, depending on whether the input signal has had a sample time hit yet during the simulation.

The initial output value is the value of the **Value until first sample time hit** parameter. This value is in effect strictly before the first sample time hit of the input signal.

## Dialog Box



**Value until first sample time hit**

The value to output before the first sample time hit of the input signal. The value of this parameter must have the same dimensions, data type, and complexity as the input signal.

**See Also**

IC, Signal Latch (Obsolete)

**Introduced in R2008a**

# Input Switch (Obsolete)

Accept entities from selected entity input port

## Library

Routing




---

**Note:** This page is the block reference page for the Input Switch block introduced before R2016a. To see the documentation on the new Input Switch block, see Entity Input Switch.

---

This block selects exactly one entity input port for potential arrivals. The selected entity input port can change during the simulation. When one entity input port becomes selected, all others become unavailable.

The rules the block uses for selecting an entity input port are listed in the table.

Switching criterion Value	Description
Round robin	At the beginning of the simulation, <b>IN1</b> is selected. After each departure, the block selects the entity input port next to the last selected port. After exhausting all entity input ports, the block returns to the first one, <b>IN1</b> .
Equiprobable	At the beginning of the simulation and after each departure, the block randomly chooses which entity input port is selected for the next arrival. All entity input ports are equally likely. The <b>Initial seed</b> parameter initializes the random number generation process.
From signal port p	Selecting this option creates an additional signal input port, labeled <b>p</b> . The signal at this port must have integer

Switching criterion Value	Description
	values between 1 and the <b>Number of entity input ports</b> parameter value. The block detects changes in this integer value and selects the corresponding entity input port for future arriving entities.

---

**Tip** If multiple entity input ports of the **Input Switch** block are on entity paths that come from a single block having multiple entity output ports, include a storage block in each path.

For example, instead of connecting two entity output ports of an **Entity Splitter** block directly to two entity input ports of an **Input Switch** block, insert a storage block in each of the two paths.

---

## Ports

### Entity Input Ports

Label	Description
IN1, IN2, IN3, and so on	Ports for potential entity arrivals. At any given time, one input port is selected and the others are unavailable. The <b>Number of entity input ports</b> parameter determines how many of these entity input ports the block has.

### Signal Input Ports

Label	Description
p	Index of the entity input port that is available. Values are 1, 2, 3,..., <b>Number of entity input ports</b> . This signal must be an event-based signal. You see this port only if you set <b>Switching criterion</b> to From signal port p.

### Entity Output Ports

Label	Description
OUT	Port for departing entities.

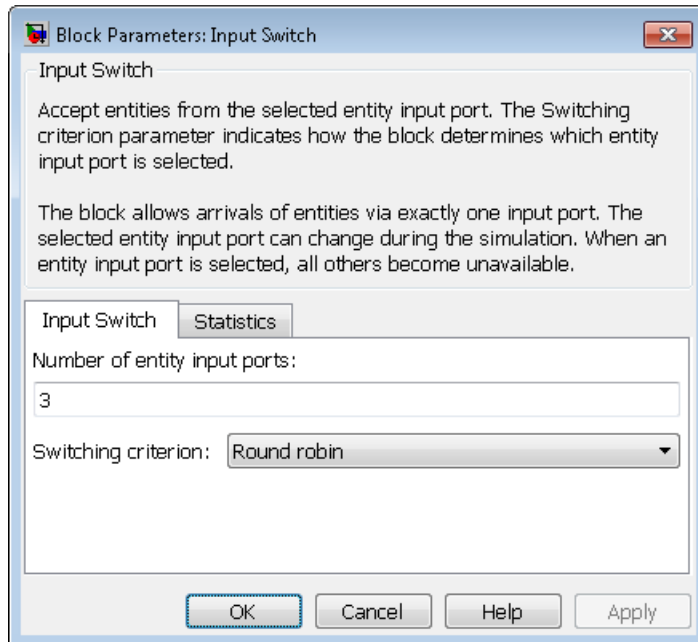
**Signal Output Ports**

<b>Label</b>	<b>Description</b>	<b>Time of Update When Statistic Is On</b>	<b>Order of Update</b>
<b>#d</b>	Number of entities that have departed from this block since the start of the simulation.	After entity departure	2
<b>last</b>	Index of the input port that was available the last time an entity departed. The initial value is 0. After an entity has departed, values are 1, 2, 3,..., <b>Number of entity input ports</b> .	After entity departure	1

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

### Input Switch Tab



#### Number of entity input ports

Determines how many entity input ports the block has.

#### Switching criterion

The rule that determines which entity input port is selected for receiving entities.

#### Initial seed

A nonnegative integer that initializes the random number generator used to select an entity input port. You see this field only if you set **Switching criterion** to Equiprobable.

#### Resolve simultaneous signal updates according to event priority

Select this option to prioritize the port-selection event explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has

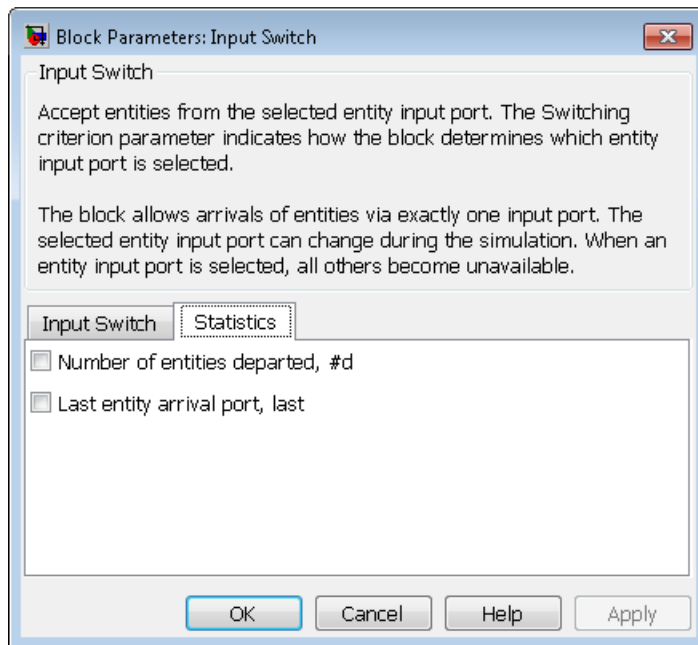
priority SYS1 on the event calendar. You see this field only if you set **Switching criterion** to From signal port p.

### Event priority

The priority of the port-selection event, relative to other simultaneous events in the simulation. You see this field only if you set **Switching criterion** to From signal port p and select **Resolve simultaneous signal updates according to event priority**.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see the preceding table, “Signal Output Ports”.



### Number of entities departed

Allows you to use the signal output port labeled **#d**.

### Last entity arrival port

Allows you to use the signal output port labeled **last**.

## **See Also**

Output Switch (Obsolete)

**Introduced before R2006a**



# Instantaneous Entity Counting Scope (Obsolete)

Plot entity count versus time

## Library

SimEvents Sinks



This block creates a plot by counting arriving entities at each arrival time. The block restarts the count from 1 when the time changes. As a result, the count is cumulative for a given time instant but not cumulative across the entire simulation.

---

**Note:** If you want to plot the total number of arriving entities across the entire simulation, connect the **#d** signal of the **Entity Departure Counter (Obsolete)** block to the **Signal Scope (Obsolete)** block.

---

Use the **Enable entity OUT port** option to choose whether the entity advances to a subsequent block or whether the block absorbs the arriving entity.

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities, which the block counts.

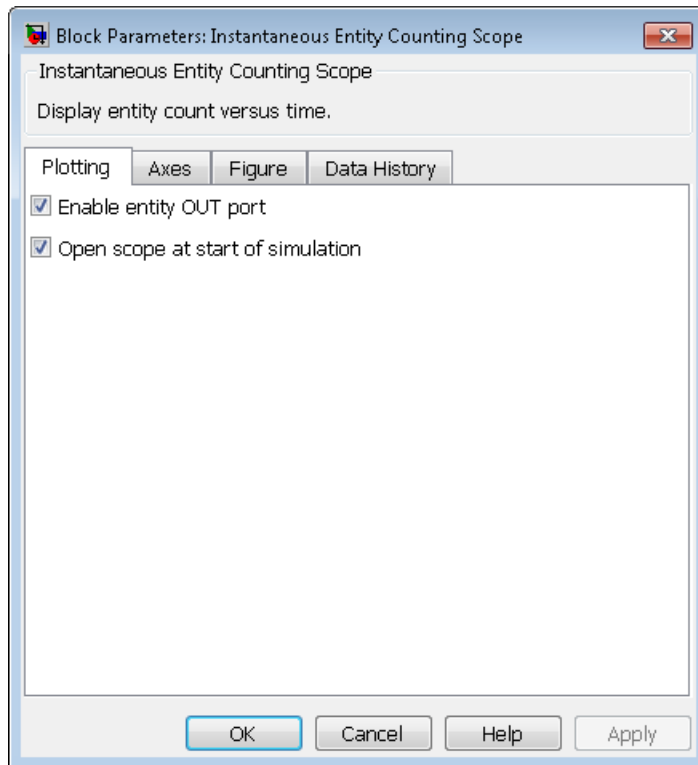
### Entity Output Ports

Label	Description
OUT	Port for departing entities. You see this port only if you select <b>Enable entity OUT port</b> .

## Dialog Box

To open the block dialog box, click the Parameters toolbar button in the plot window.

## Plotting Tab



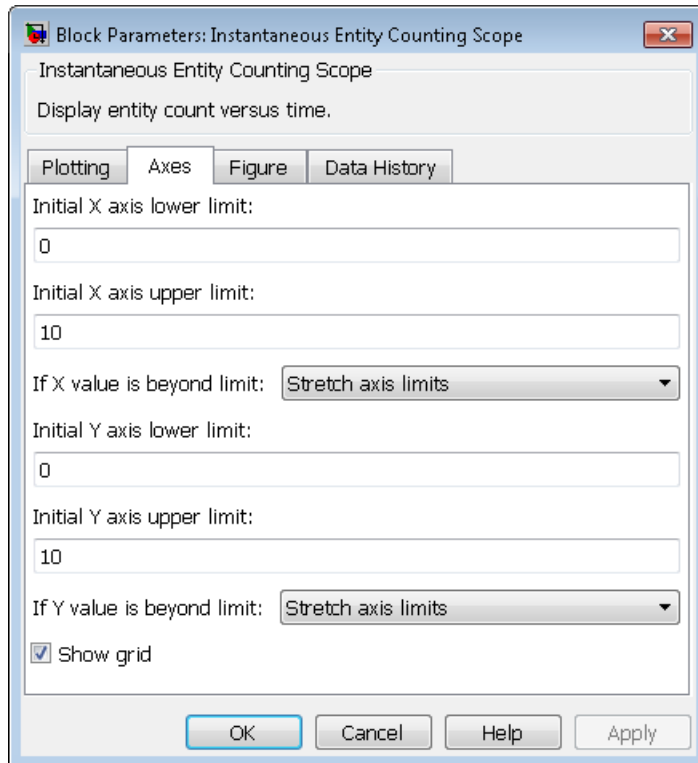
### Enable entity OUT port

Causes the block to have an entity output port labeled **OUT**, through which the arriving entity departs. If you clear this box, the block absorbs arriving entities.

### Open scope at start of simulation

Selecting this option causes the plot window to open when you start the simulation. If you clear this box, you can open the plot window by double-clicking the block icon.

## Axes Tab



### Initial X axis lower limit, Initial X axis upper limit

The interval shown on the X axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If X value is beyond limit** setting.

### If X value is beyond limit

Determines how the plot changes if one or more X values are not within the limits shown on the X axis.

### Initial Y axis lower limit, Initial Y axis upper limit

The interval shown on the Y axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If Y value is beyond limit** setting.

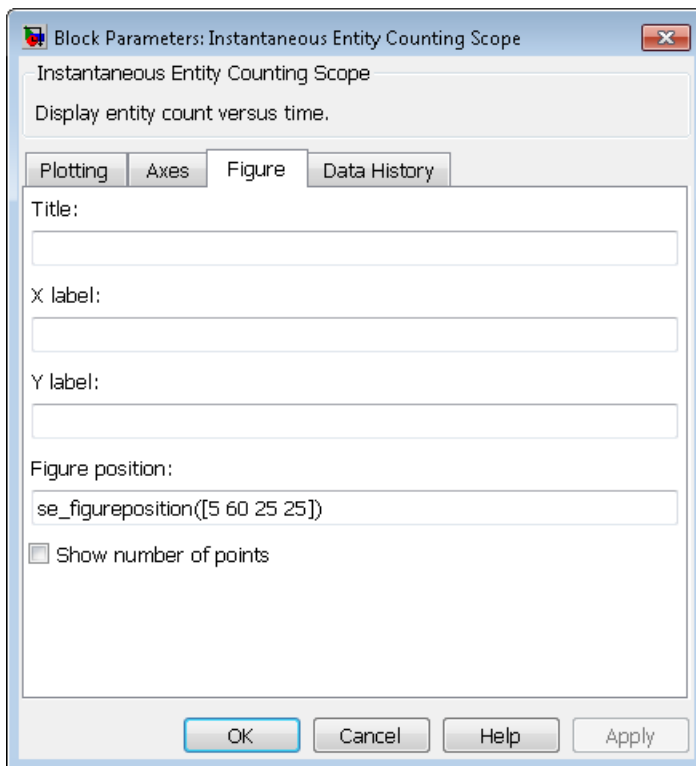
**If Y value is beyond limit**

Determines how the plot changes if one or more entity counts are not within the limits shown on the Y axis.

**Show grid**

Toggles the grid on and off.

**Figure Tab**



**Title**

Text that appears as the title of the plot, above the axes.

**Y label**

Text that appears to the left of the vertical axis.

**X label**

Text that appears below the horizontal axis.

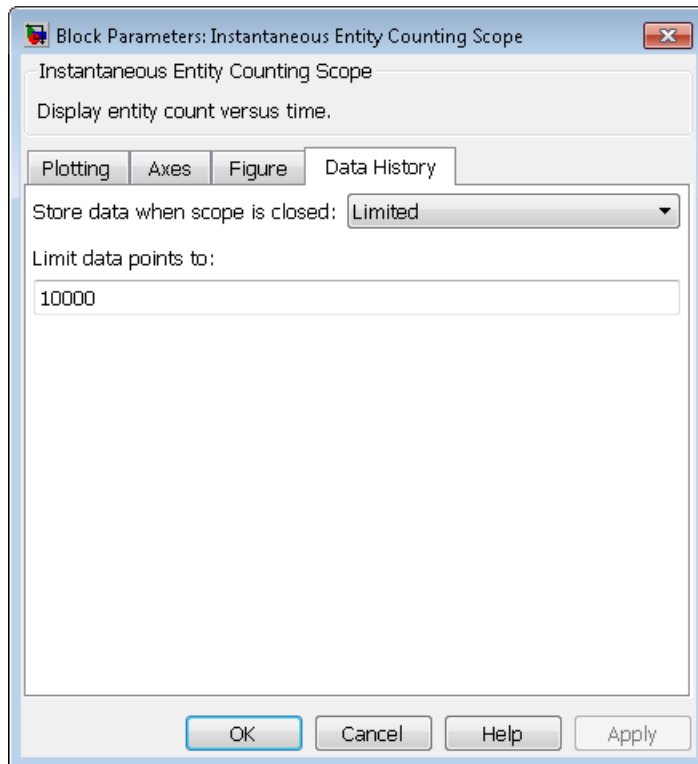
### Position

A four-element vector of the form [left bottom width height] specifying the position of the scope window. (0,0) is the lower left corner of the display.

### Show number of entities

Displays the number of plotted points using an annotation in the plot window.

## Data History Tab



### Store data when scope is closed

Select **Unlimited** to have the block cache all data for future viewing, **Limited** to cache a portion of the most recent data, and **Disabled** to avoid caching undisplayed data.

**Limit data points to**

The number of data points the block caches, using the most recent data. You see this field only if you set **Store data when scope is closed** to **Limited**.

**See Also**

Entity Departure Counter (Obsolete), Instantaneous Event Counting Scope (Obsolete)

“Count Entities”

**Introduced before R2006a**

# Instantaneous Event Counting Scope (Obsolete)

Plot event count versus time

## Library

SimEvents Sinks

## Description



This block creates a plot by counting events. The block restarts the count from 1 when the time changes. As a result, the count is cumulative for a given time instant but not cumulative across the entire simulation.

When the block has a **ts** input port and the input signal is an event-based signal, a stem with no marker represents the initial output of the signal.

## Ports

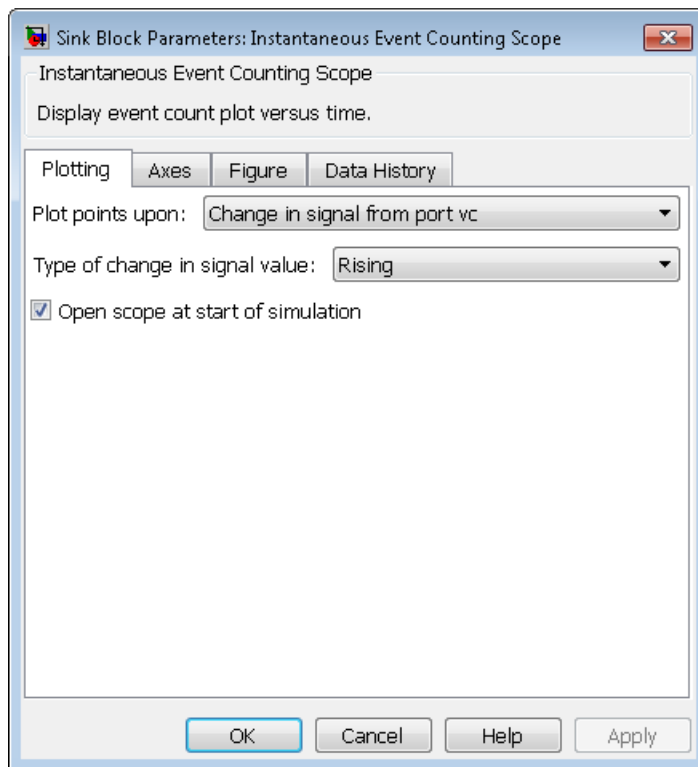
### Signal Input Ports

Label	Description
<b>ts</b>	When this signal has an update, the counter increments. This signal must be an event-based signal. You see this port only if you set <b>Plot points upon</b> to <b>Sample time hit</b> from port <b>ts</b> .
<b>tr</b>	When this signal satisfies the specified trigger criteria, the counter increments. This signal must be an event-based signal. You see this port only if you set <b>Plot points upon</b> to <b>Trigger</b> from port <b>tr</b> .
<b>vc</b>	When this signal satisfies the specified value-change criteria, the counter increments. This signal must be an event-based signal. You see this port only if you set <b>Plot points upon</b> to <b>Change in signal</b> from port <b>vc</b> .
<b>fcn</b>	When this signal carries a function call, the counter increments. This signal must be an event-based function call. You see this port only if you set <b>Plot points upon</b> to <b>Function call</b> from port <b>fcn</b> .

## Dialog Box

To open the block dialog box, click the Parameters toolbar button in the plot window.

### Plotting Tab



#### Plot points upon

The type of event that indicates when the block increments its counter.

#### Trigger type, Type of change in signal value

**Trigger type** determines whether rising, falling, or either type of trigger edge causes the block to increment its counter. You see this field only if you set **Plot points upon** to Trigger from port tr.

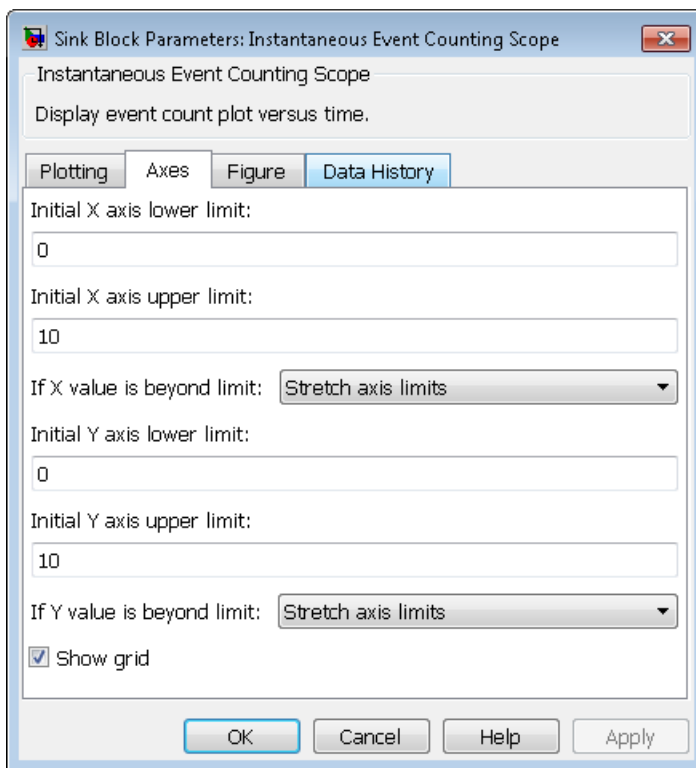


**Type of change in signal value** determines whether rising, falling, or either type of value change causes the block to increment its counter. You see this field only if you set **Plot points upon** to Change in signal from port vc.

### Open scope at start of simulation

Selecting this option causes the plot window to open when you start the simulation. If you clear this box, you can open the plot window by double-clicking the block icon.

## Axes Tab



### Initial X axis lower limit, Initial X axis upper limit

The interval shown on the X axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If X value is beyond limit** setting.

### If X value is beyond limit

Determines how the plot changes if one or more X values are not within the limits shown on the X axis.

### **Initial Y axis lower limit, Initial Y axis upper limit**

The interval shown on the Y axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If Y value is beyond limit** setting.

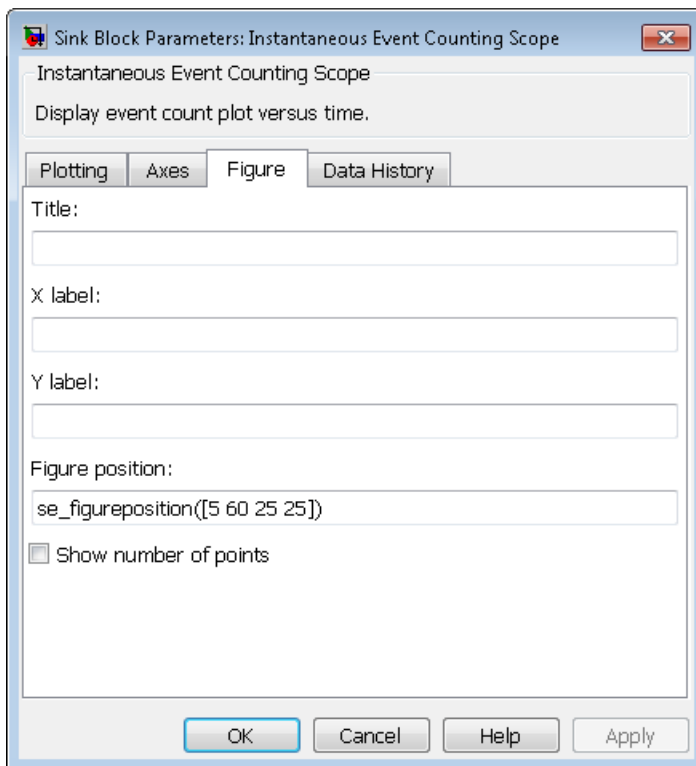
### **If Y value is beyond limit**

Determines how the plot changes if one or more event counts are not within the limits shown on the Y axis.

### **Show grid**

Toggles the grid on and off.

## **Figure Tab**



**Title**

Text that appears as the title of the plot, above the axes.

**Y label**

Text that appears to the left of the vertical axis.

**X label**

Text that appears below the horizontal axis.

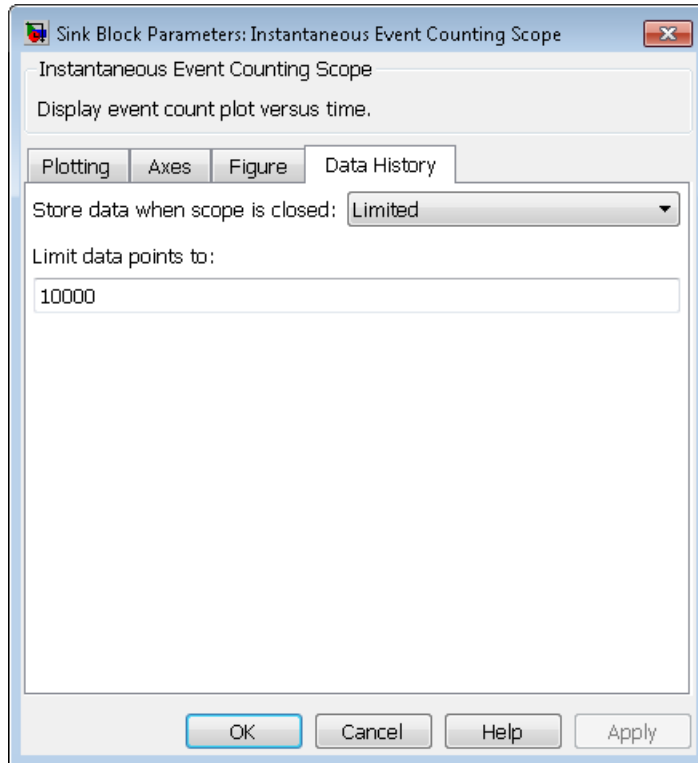
**Position**

A four-element vector of the form [left bottom width height] specifying the position of the scope window. (0,0) is the lower left corner of the display.

**Show number of points**

Displays the number of plotted points using an annotation in the plot window.

## Data History Tab



### Store data when scope is closed

Select **Unlimited** to have the block cache all data for future viewing, **Limited** to cache a portion of the most recent data, and **Disabled** to avoid caching undisplayed data.

### Limit data points to

The number of data points the block caches, using the most recent data. You see this field only if you set **Store data when scope is closed** to **Limited**.

## See Also

Signal Scope (Obsolete), Instantaneous Entity Counting Scope (Obsolete)

**Introduced before R2006a**

## LIFO Queue (Obsolete)

Store entities in stack for undetermined length of time

### Library

Queues



This block stores up to  $N$  entities simultaneously, where  $N$  is the **Capacity** parameter value. The block attempts to output an entity through the **OUT** port but retains the entity if the **OUT** port is blocked. If the block is storing multiple entities and no entity times out, then entities depart in a last-in, first-out (LIFO) fashion. If an entity in this block is scheduled to time out, then it might depart prematurely via the optional **TO** port. The length of time that an entity stays in this block cannot be determined in advance.

The **IN** port is unavailable whenever this block stores exactly  $N$  entities. In this case, the queue is said to be full.

### Ports

#### Entity Input Ports

Label	Description
IN	Port for arriving entities, which will be stored.

#### Entity Output Ports

Label	Description
OUT	Port for departing entities that do not time out while in this block.

Label	Description
<b>TO</b>	Port for entities that time out while in this block. You see this port only if you select <b>Enable TO port for timed-out entities</b> . This port must not be blocked when an entity attempts to depart here.

### Signal Output Ports

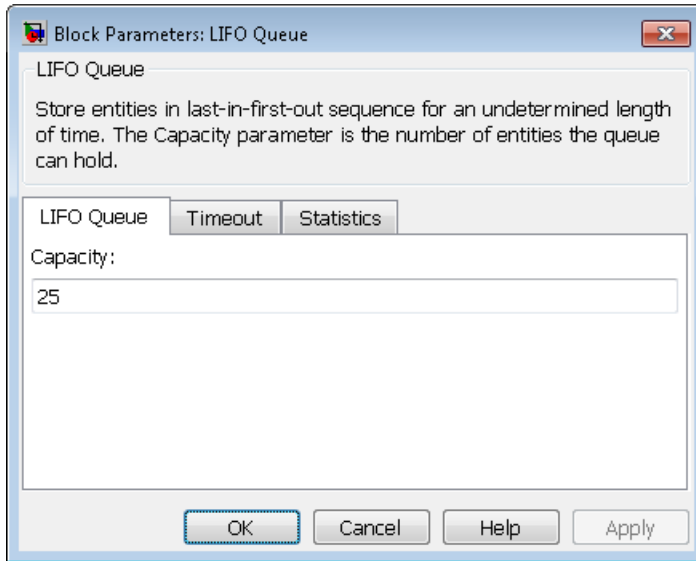
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
<b>#d</b>	Number of entities that have departed from this block via the <b>OUT</b> port since the start of the simulation.	After entity departure via the <b>OUT</b> port	3
<b>#n</b>	Number of entities currently in the queue.	After entity arrival and after entity departure	2
<b>w</b>	Sample mean of the waiting times in this block for all entities that have departed via any port.	After entity departure	1
<b>len</b>	Average number of entities in the queue over time, that is, the time average of the <b>#n</b> signal.	After entity arrival and after entity departure.	1
<b>#to</b>	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the <b>TO</b> port	3

Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

### LIFO Queue Tab

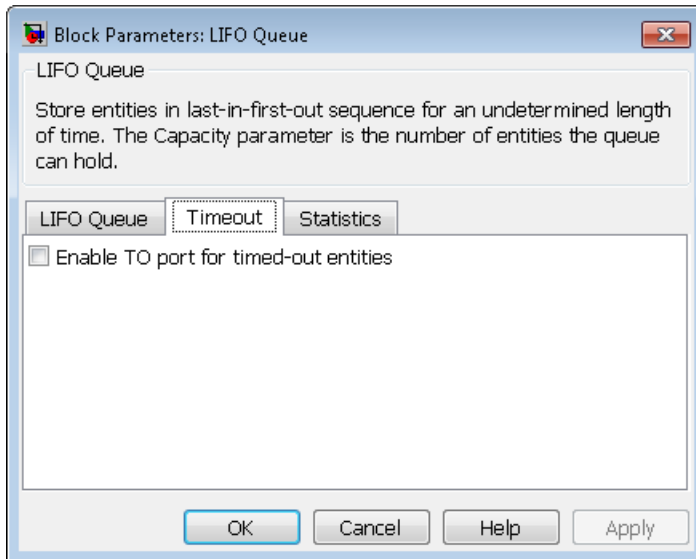


### Capacity

Determines how many entities the block can store at a time. The capacity must be a positive integer or Inf.



## Timeout Tab

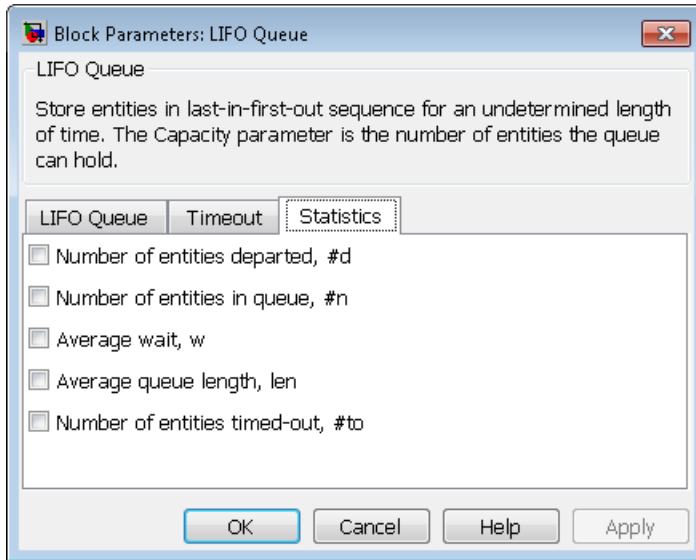


### Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the **Schedule Timeout (Obsolete)** block.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see the preceding table, “Signal Output Ports”.



### **Number of entities departed**

Allows you to use the signal output port labeled **#d**.

### **Number of entities in queue**

Allows you to use the signal output port labeled **#n**.

### **Average wait**

Allows you to use the signal output port labeled **w**.

### **Average queue length**

Allows you to use the signal output port labeled **len**.

### **Number of entities timed out**

Allows you to use the signal output port labeled **#to**.

## **See Also**

FIFO Queue (Obsolete), Priority Queue (Obsolete)

**Introduced before R2006a**

# N-Server (Obsolete)

Serve up to N entities for period of time

## Library

Servers



This block stores up to N entities, serving each one independently for a period of time and then attempting to output the entity through the **OUT** port. If the **OUT** port is blocked, then the entity stays in this block until the port becomes unblocked. If an entity in this block is scheduled to time out, then it might depart prematurely via the optional **TO** port.

An N-server is like a set of N single servers connected in parallel, followed by a path combiner; the path combiner notifies entities of an unblocked path in the sequence in which the entities completed their service time, until one entity departs.

You specify the service time, which is the duration of service, via a parameter, attribute, or signal, depending on the **Service time from** parameter value. The block determines the service time for an entity upon its arrival. Service times are assumed to be specified in seconds.

---

**Note:** If you specify the service time via an event-based signal, be sure that its updates occur before the entity arrives.

---

All entities that arrive do so via the **IN** port. The **IN** port is unavailable whenever this block contains N entities. In that case, the **IN** port becomes available when at least one of the N entities departs.

## Ports

### Entity Input Ports

Port Label	Description
IN	Port for arriving entities, which will be served.

### Signal Input Ports

Port Label	Description
t	Service time, in seconds, for a newly arrived entity. This signal must be an event-based signal. You see this port only if you set <b>Service time from</b> to <b>Signal port t</b> .
pause	Port for input signal that disables all servers when the signal is positive. While the servers are disabled, any occupied servers retain their entities and the software pauses the remaining service time for each server. When the signal at the input port becomes nonpositive, each server resumes service. You see this port only if you select <b>Allow service control</b> and set <b>Service change upon disabling</b> to <b>Pause</b> .
complete	<p>Port for input signal that disables all servers when the signal is positive.</p> <p>When a positive signal enters the <b>complete</b> port, the software:</p> <ul style="list-style-type: none"> <li>• Disables all servers.</li> <li>• Immediately completes service in all occupied servers.</li> <li>• Resets the remaining service time in all servers.</li> </ul> <p>If no blockage exists at the entity output port of the <b>N-Server</b> block, entities immediately advance from occupied servers to downstream blocks. When the signal at the input port becomes nonpositive, normal behavior of the <b>N-Server</b> block resumes.</p> <p>You see this port only if you select <b>Allow service control</b> and set <b>Service change upon disabling</b> to <b>Force complete</b>.</p>

### Entity Output Ports

Port Label	Description
OUT	Port for departing entities that have completed their service time and have not timed out while in this block.

Port Label	Description
TO	Port for entities that time out while in this block. You see this port only if you select <b>Enable TO port for timed-out entities</b> . This port must not be blocked when an entity attempts to depart here.

### Signal Output Ports

Port Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block via the <b>OUT</b> port since the start of the simulation.	After entity departure via the <b>OUT</b> port	5
#n	Number of entities currently in the block, between 0 and N.	After entity arrival and after entity departure	4
pe	A value of 1 indicates that the block stores at least one entity that has tried and failed to depart. Such entities are pending entities.  A value of 0 indicates that the block does not store any pending entities.	After the block stores an entity that has tried and failed to depart. In this case, the signal value is 1.  After the departure of a pending entity. In this case, the signal value depends on whether any other pending entities remain in the block.	1
#pe	Number of pending entities in the block.	After the block stores an entity that has tried and failed to depart.  After the departure of a pending entity.	3
w	Sample mean of the waiting times in this block for all entities that have departed via any port. An	After entity departure	2

Port Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
	entity's waiting time might exceed its service time if the <b>OUT</b> port is blocked when the entity completes service.		
<b>util</b>	Utilization of the N-server. If <b>Number of servers</b> is finite, <b>util</b> is the time average of the fraction of servers that are storing an entity. At time values when an entity arrives or departs, <b>util</b> equals $1/N$ times the time average of the <b>#n</b> signal. If <b>Number of servers</b> is infinite, then <b>util</b> is always zero.	Performance considerations cause the block to update the signal only after each arrival or departure of an entity.	2
<b>#to</b>	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the <b>TO</b> port	5
<b>so</b>	Occupancy status of each server in the N-Server block. The <b>so</b> port outputs a vector of values. If a server is unoccupied, the value of the corresponding vector element is <b>0</b> . If a server is occupied, the vector element has a value of <b>1</b> .	After entity arrival and after entity departure	6

Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

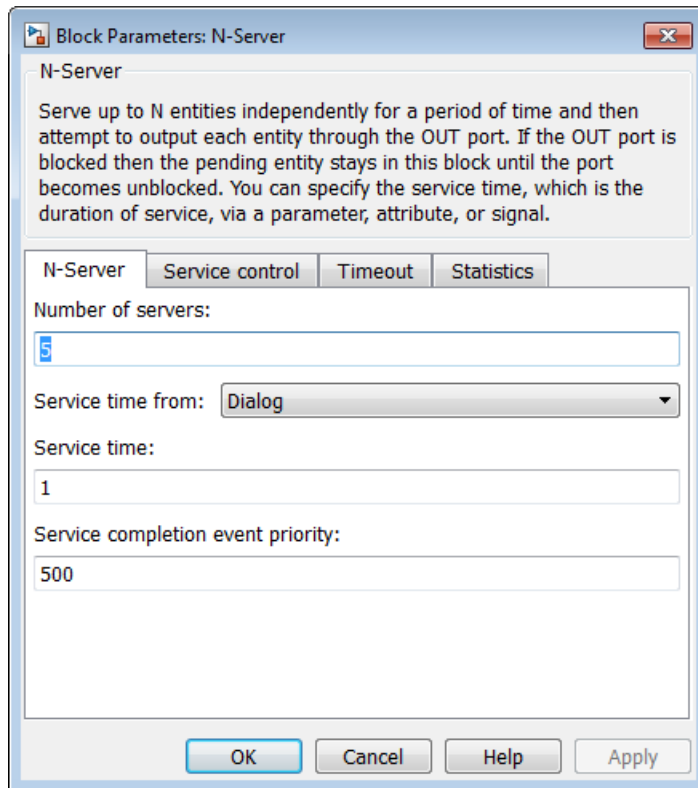
A more precise definition of the utilization signal **util** at an update time  $T > 0$  is

$$\frac{1}{T} \sum_k \left( \frac{(\#n)_k}{N} \right) \cdot \text{length}(I_k)$$

where  $I_k$  is the  $k$ th time interval between successive pairs of times that **util** is updated and  $(\#n)_k$  is the number of entities the N-Server block is storing during the open interval  $I_k$ . If an update of **util** occurs at  $T=0$ , the value is  $\#n/N$ .

## Dialog Box

### N-Server Tab



#### Number of servers

The number of servers the block represents, N.

#### Service time from

Determines whether the service time is computed from a parameter in this dialog box, an input signal, or an attribute of the entity being served.

#### Service time



The service time, in seconds, for all entities. You see this field only if you set **Service time from** to **Dialog**.

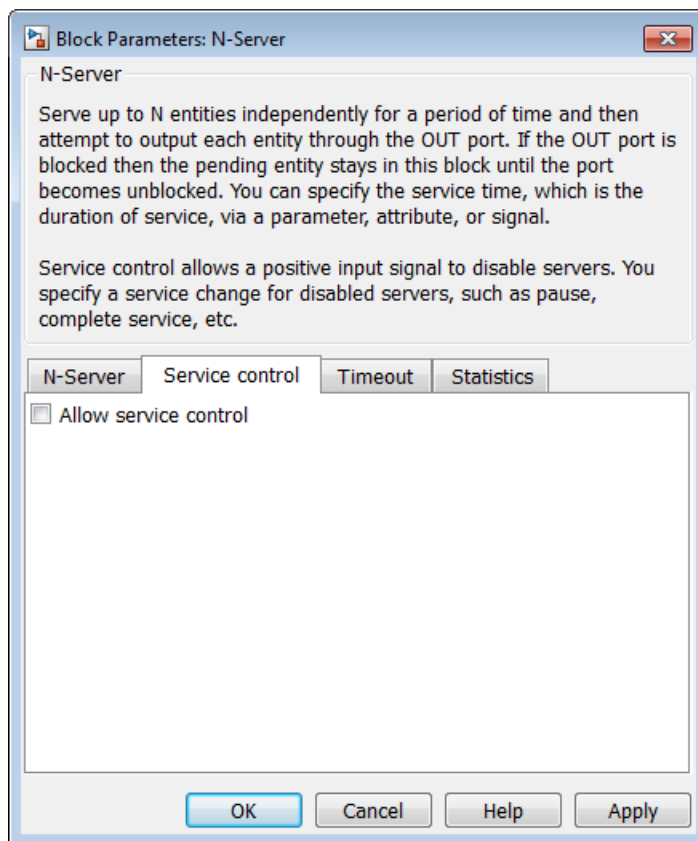
#### Attribute name

The name of the attribute whose value the block uses as the service time for an entity. You see this field only if you set **Service time from** to **Attribute**.

#### Service completion event priority

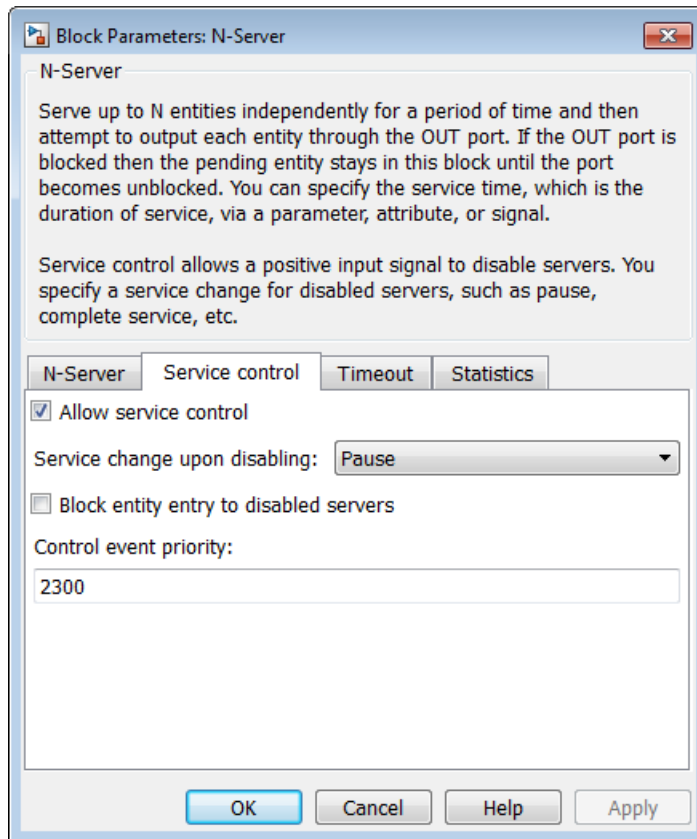
The priority of the service completion event, relative to other simultaneous events in the simulation.

## Service control Tab



### Allow service control

Adds an input signal port to the block. When you input a positive signal to this added signal port, the software disables servers in the block and applies a service change. You specify the service change action using an option that becomes visible when you select **Allow service control**.



### Service change upon disabling

Specifies the service change action that the software applies to disabled servers. You see this option only if you select **Allow service control**.

By default, **Service change upon disabling** is set to **Pause**. When **Pause** is selected, the signal input port added by **Allow service control** is labeled **pause**.

When you input a positive signal to the **pause** port, the software disables all servers in the block. While this input signal remains positive, any occupied servers retain their entities and the software pauses the remaining service time for each server. When the signal at the input port becomes nonpositive, each server resumes service.

You can also set **Service change upon disabling** to **Force complete**. In this case, when you click **OK**, the label of the signal input port added by **Allow service control** changes to **complete**.

When a positive signal enters the **complete** port, the software:

- Disables all servers.
- Immediately completes service in all occupied servers.
- Resets the remaining service time in all servers.

If no blockage exists at the entity output port of the **N-Server** block, entities immediately advance from occupied servers to downstream blocks. When the signal at the input port becomes nonpositive, normal behavior of the **N-Server** block resumes.

For an example of using the **complete** port, see “Task Preemption in a Multitasking Processor”.

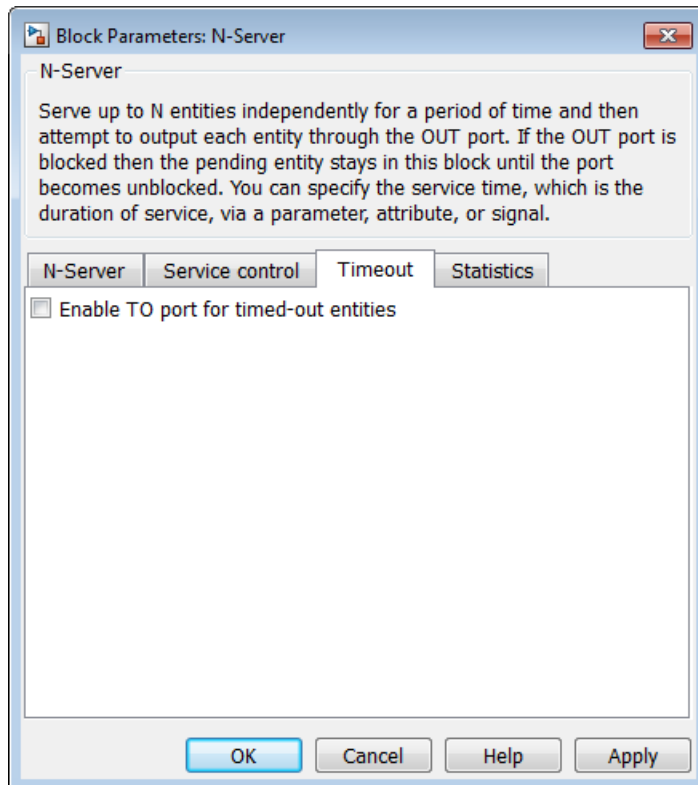
### **Block entity entry to disabled servers**

Determines whether unoccupied servers can accept entities while the signal at the **pause** or **complete** port is positive.

### **Control event priority**

The priority of the service control event, relative to other simultaneous events in the simulation.

### Timeout Tab

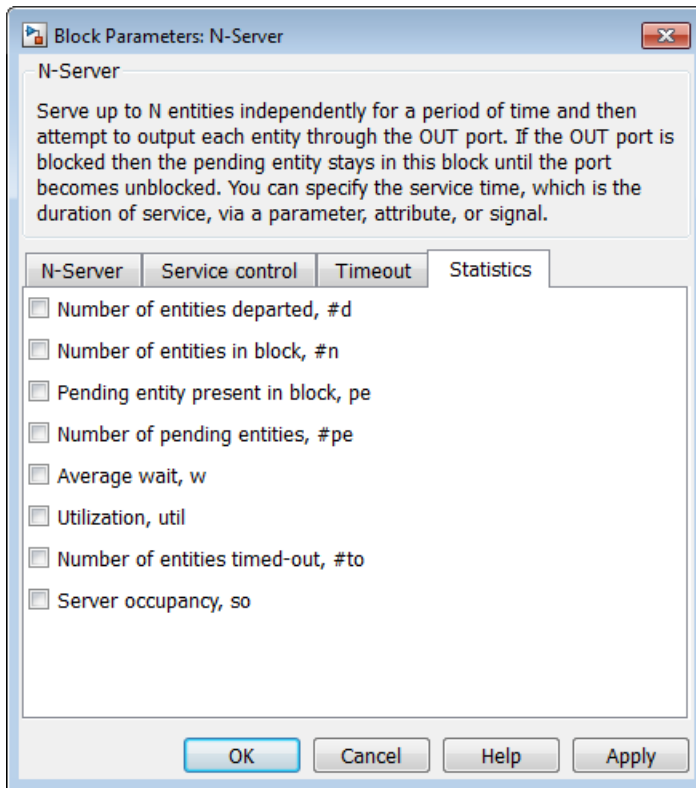


#### Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the **Schedule Timeout (Obsolete)** block.

### Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see the preceding table, "Signal Output Ports".



### **Number of entities departed**

Allows you to use the signal output port labeled **#d**.

### **Number of entities in block**

Allows you to use the signal output port labeled **#n**.

### **Pending entity present in block**

Allows you to use the signal output port labeled **pe**.

### **Number of pending entities**

Allows you to use the signal output port labeled **#pe**.

### **Average wait**

Allows you to use the signal output port labeled **w**.

### **Utilization**

Allows you to use the signal output port labeled **util**.

**Number of entities timed out**

Allows you to use the signal output port labeled **#to**.

**Server occupancy, so**

Allows you to use the signal output port labeled **so**.

## **See Also**

Single Server (Obsolete), Infinite Server (Obsolete)

**Introduced before R2006a**

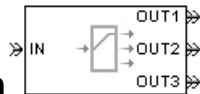
# Output Switch (Obsolete)

Select entity output port for departure

## Library

Routing

## Description



---

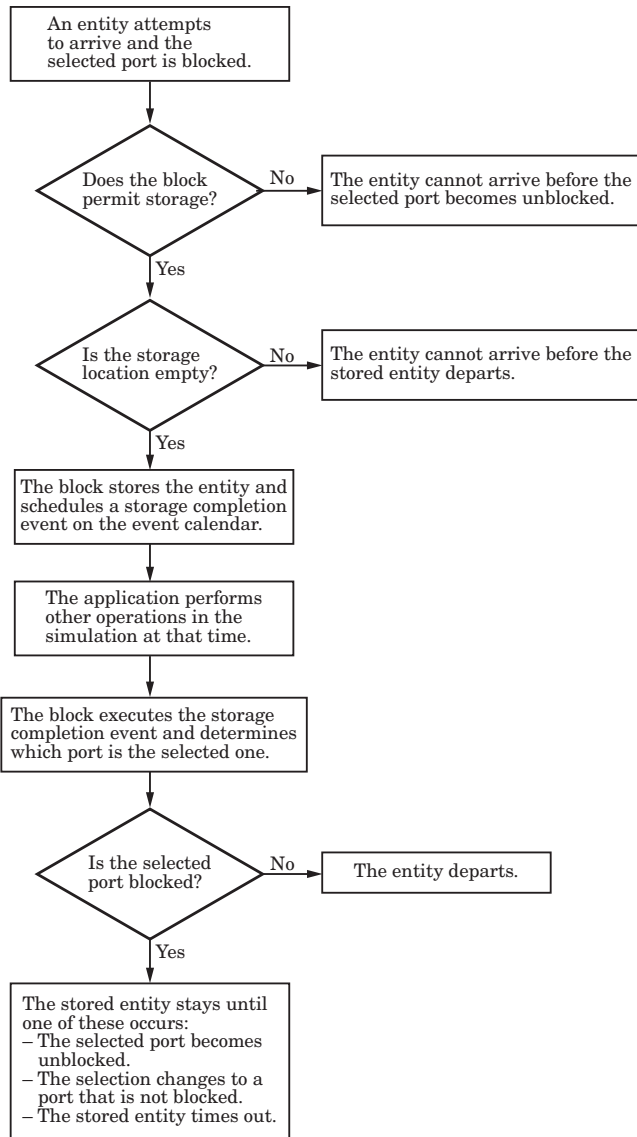
**Note:** This page is the block reference page for the Output Switch block introduced before R2016a. To see the documentation for the new Output Switch block, see Entity Output Switch.

---

This block receives entities, which depart through one of multiple entity output ports. The selected port can change during the simulation.

## Managing Arrivals and Departures

When the selected port is not blocked, an arriving entity departs through that port. When an entity attempts to arrive and the selected port is blocked, the block's behavior depends on the block's configuration and state, as illustrated in the figure.



---

**Note:** This block permits storage only if you set **Switching criterion** to From signal port `p`, and then select **Store entity before switching**.

---



Entities that time out depart via the block's **TO** port.

## Switching Criteria

The **Switching criterion** parameter indicates how the block determines which entity output port is selected for departure at any given time. The values of the **Switching criterion** parameter are described in the table below.

Switching criterion Value	Description
Round robin	The first arriving entity in the simulation departs via the <b>OUT1</b> port. Upon each subsequent arrival, the block selects the entity output port next to the last selected port. After exhausting all entity output ports, the block returns to the first one, <b>OUT1</b> .
Equiprobable	At the beginning of the simulation and upon each departure, the block randomly chooses the entity output port through which the next arriving entity departs. All entity output ports are equally likely to be selected. The <b>Initial seed</b> parameter initializes the random number generation process.
First port that is not blocked	When an entity attempts to arrive, the block attempts to output the entity through <b>OUT1</b> . If that port is blocked, then the block attempts to output the entity through <b>OUT2</b> , and so on. If all entity output ports are blocked, then this block's <b>IN</b> port is unavailable and the entity cannot arrive.
From signal port p	Selecting this option creates an additional signal input port, labeled <b>p</b> . The signal at this port uses integer values between 1 and the <b>Number of entity output ports</b> parameter value to refer to entity output ports. The block monitors the <b>p</b> signal's value throughout the simulation and reacts to changes by selecting the corresponding entity output port.

Switching criterion Value	Description
From attribute	An arriving entity departs through the entity output port that corresponds to the value of an attribute of your choice. Name the attribute using the <b>Attribute name</b> parameter. The attribute value must be an integer between 1 and the <b>Number of entity output ports</b> parameter value. If the indicated entity output port is blocked, then this block does not accept the entity for arrival until the entity output port becomes unblocked.

---

**Note:** If you set **Switching criterion** to **From signal port p**, then the block offers several options to help you ensure that the signal is up to date and valid when the block uses it to determine how to process the arriving entity. Be especially careful when the signal is in a feedback loop, or when the signal can change at the same time an entity arrives. For details, see “Use Messages To Route Entities”.

---

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities.

### Signal Input Ports

Label	Description
p	Index of the entity output port through which an arriving entity departs. Values must be integers between 1 and <b>Number of entity output ports</b> . This signal must be an event-based signal. You see this port only if you set <b>Switching criterion</b> to <b>From signal port p</b> .

### Entity Output Ports

Label	Description
OUT1, OUT2,	Entity ports through which an arriving entity departs, where the <b>Switching criterion</b> parameter determines which of multiple ports the entity departs

Label	Description
OUT3, and so on	through. The <b>Number of entity output ports</b> parameter determines how many of these entity output ports the block has.
TO	Port for entities that time out while in this block. You see this port only if you set <b>Switching criterion</b> to From signal port p, select <b>Store entity before switching</b> , and select <b>Enable TO port for timed-out entities</b> . This port must not be blocked when an entity attempts to depart here.

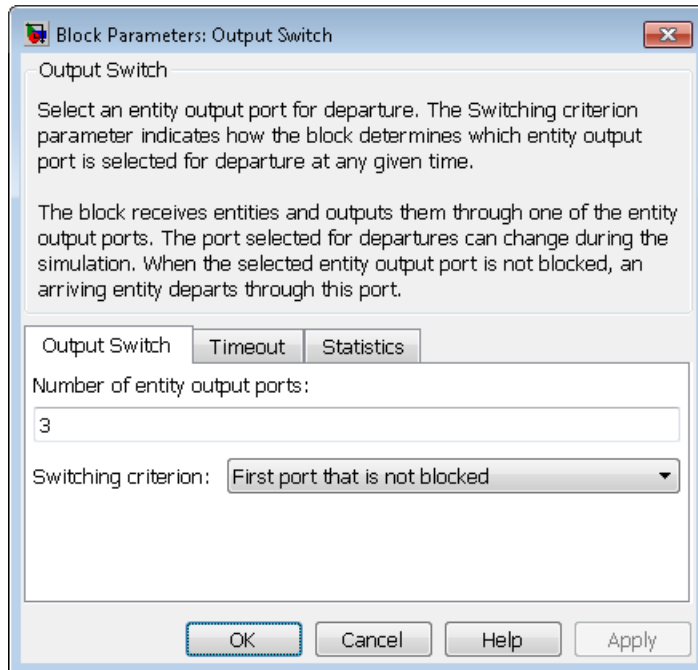
Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update
<b>#d</b>	Number of entities that have departed from this block without timing out, since the start of the simulation.	After entity departure via a port other than <b>TO</b>	3
<b>#to</b>	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the <b>TO</b> port	2
<b>pe</b>	A value of 1 indicates that the block stores an entity that has tried and failed to depart. In that case, the entity is a pending entity.  A value of 0 indicates that the block does not store any pending entities.	Sample time hit of 1 occurs after the block stores an entity that has tried and failed to depart.  Sample time hit of 0 occurs after the departure of the pending entity via any port.	1
<b>last</b>	Index of the output port through which the last entity departed, excluding timed-out entities. Aside from the initial output, values of this signal are 1, 2, 3,..., <b>Number of entity output ports</b> .	After entity departure via a port other than <b>TO</b>	2

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

### Output Switch Tab



#### Number of entity output ports

Determines how many entity output ports the block has.

#### Switching criterion

The rule that determines which entity output port an arriving entity departs through.

#### Initial seed

A nonnegative integer that initializes the random number generator used to select an entity output port. You see this field only if you set **Switching criterion** to Equiprobable.

#### Specify initial port selection

Select this option to indicate the initially selected entity output port. For details, see “Specify an Initial Port Selection”. You see this field only if you set **Switching criterion** to From signal port *p*.

### **Initial port selection**

The entity output port that the block selects when the simulation begins. The value must be an integer between 1 and **Number of entity output ports**. The block uses **Initial port selection** instead of the *p* signal's value until the signal has its first sample time hit. You see this field only if you set **Switching criterion** to From signal port *p* and select **Specify initial port selection**.

### **Store entity before switching**

If you select this option, the block can store one entity at a time. Furthermore, the block decouples its arrival and departure processing to give other blocks in the simulation an opportunity to update the *p* signal if appropriate. If you do not select this option, the block processes an arrival and departure as an atomic operation and assumes that the *p* signal is already up to date at the given time. You see this field only if you set **Switching criterion** to From signal port *p*.

### **Resolve simultaneous signal updates according to event priority**

Select this option to prioritize the port-selection event explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority SYS1 on the event calendar. You see this field only if you set **Switching criterion** to From signal port *p* and do not select **Store entity before switching**.

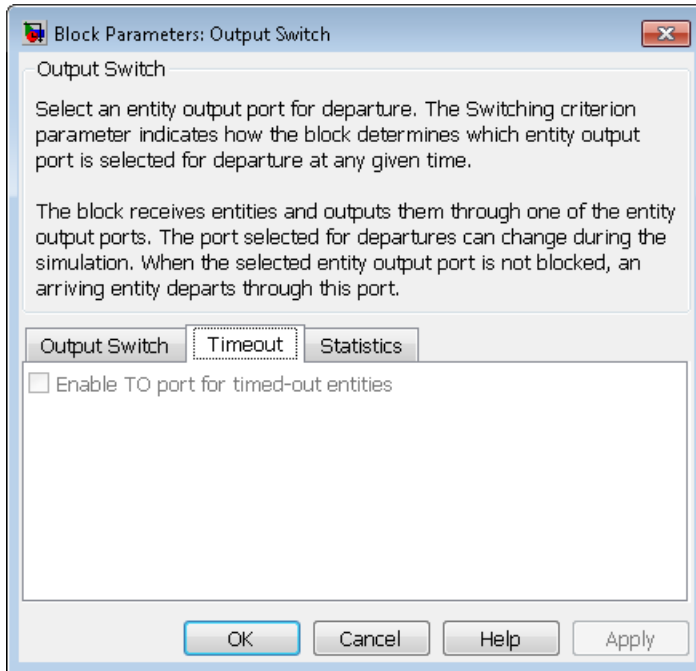
### **Event priority**

The priority of the port-selection event, relative to other simultaneous events in the simulation. **Switching criterion** to From signal port *p*, do not select **Store entity before switching**, and select **Resolve simultaneous signal updates according to event priority**.

### **Attribute name**

The name of an attribute used to select an entity output port. You see this field only if you set **Switching criterion** to From attribute.

## Timeout Tab

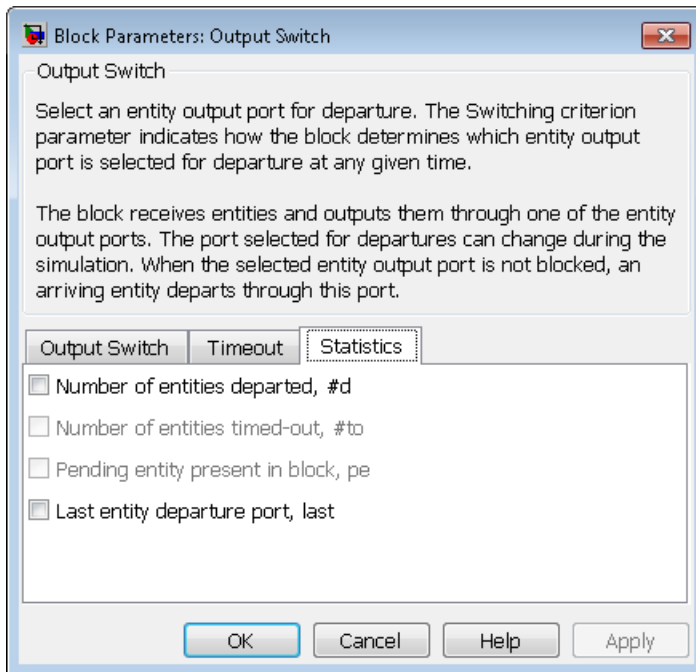


### Enable TO port for timed-out entities

This option is available only if you set **Switching criterion** to **From signal port p**, and then select **Store entity before switching** on the **Output Switch** tab of the dialog box. This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the description of the **If entity has no destination when timeout occurs** parameter on the **Schedule Timeout (Obsolete)** block reference page.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see the preceding table, “Signal Output Ports”.



### Number of entities departed

Allows you to use the signal output port labeled **#d**.

### Number of entities timed out

Allows you to use the signal output port labeled **#to**.

### Pending entity present in block, **pe**

Allows you to use the signal output port labeled **pe**. You can select this check box only if you set **Switching criterion** to **From signal port p**, and then select **Store entity before switching** on the **Output Switch** tab of the dialog box.

### Last entity departure port

Allows you to use the signal output port labeled **last**.

## See Also

Input Switch (Obsolete)



“Use Messages To Route Entities”

**Introduced before R2006a**

## Path Combiner (Obsolete)

Merge entity paths

### Library

Routing



### Description

This block accepts entities through any entity input port and outputs them through a single entity output port. You specify the number of entity input ports using the **Number of entity input ports** parameter.

If multiple entities arrive at the **Path Combiner** block simultaneously while the entity output port is not blocked, then the sequence in which the entities depart depends on the sequence of departure events from blocks that precede the **Path Combiner** block. Even if the departure time is the same for multiple entities, the sequence might affect the system's behavior. For example, if the entities advance to a queue, the departure sequence determines their positions in the queue.

Multiple instances of entities of the same type, but with different attributes, can arrive at the **Path Combiner** block. In these situations, the compiled entity type displays the union type.

### Input Port Precedence

The **Input port precedence** parameter indicates how the block determines which entity input port to notify first, whenever the entity output port changes its status from blocked to unblocked. The first notified port is the first port to become available to an arriving entity. Choices for the **Input port precedence** parameter are described in the following table.

Input Port Precedence	Action when Entity Output Port Becomes Unblocked	Example for Block with Four Entity Input Ports
IN1 port	Notify entity input ports <b>IN1, IN2, IN3</b> ,... until either an entity arrives or all ports are notified.	Throughout the simulation, the sequence of notifications is always <b>IN1, IN2, IN3, IN4</b> .
Equiprobable	Notify a random entity input port. All are equally likely and the <b>Initial seed</b> parameter initializes the random number generator. If this does not result in an entity arrival, notify the subsequent ports in turn until either an entity arrives or all ports are notified.	If the random number is three, notify the ports in the sequence <b>IN3, IN4, IN1, IN2</b> . If the random number is two on the next such occasion, notify the ports in the sequence <b>IN2, IN3, IN4, IN1</b> .
Round robin	Notify the port next to the one through which the last departing entity arrived. The <b>IN1</b> port is considered “next to” the last entity input port on the block. If this does not result in an entity arrival, notify the subsequent ports in turn until either an entity arrives or all ports are notified.	An entity arrives through the <b>IN2</b> port and advances to a <b>Single Server</b> block. Meanwhile, entities attempt to arrive at the <b>Path Combiner</b> block. When the server becomes available, the <b>Path Combiner</b> block notifies the ports in the sequence <b>IN3, IN4, IN1, IN2</b> . The sequence starts with <b>IN3</b> because it is next to <b>IN2</b> , which is the port through which the last departing entity arrived.
From signal port <b>p</b>	Notify the port whose index is the value of the <b>p</b> input signal. If this does not result in an entity arrival, notify the subsequent ports in turn until either an entity arrives or all ports are notified.	If the value of the <b>p</b> signal is three, notify the ports in the sequence <b>IN3, IN4, IN1, IN2</b> . If <b>p</b> is two on the next such occasion, notify the ports in the sequence <b>IN2, IN3, IN4, IN1</b> .

## Ports

### Entity Input Ports

Label	Description
IN1, IN2, IN3, and so on	Port for arriving entities. The <b>Number of entity input ports</b> parameter determines how many of these entity input ports the block has.

### Signal Input Ports

Label	Description
<b>p</b>	Index of the entity input port that the block makes available first, upon an event that changes the entity output port from blocked to unblocked. Values are 1, 2, 3,..., <b>Number of entity input ports</b> . This signal must be an event-based signal. You see this port only if you set <b>Input port precedence</b> to From signal port p.

### Entity Output Ports

Label	Description
<b>OUT</b>	Port for departing entities.

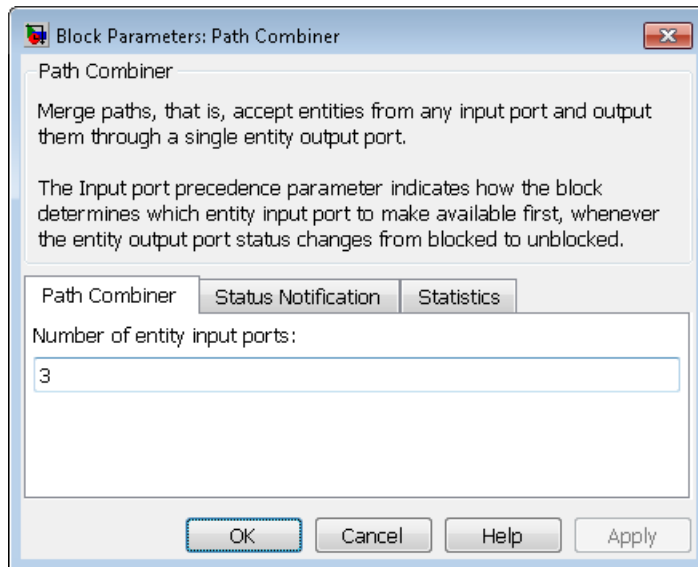
### Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update
<b>#d</b>	Number of entities that have departed from this block since the start of the simulation.	After entity departure	2
<b>last</b>	Index of the input port through which the last entity arrived. The initial value is 0. After an entity has arrived and departed, values are 1, 2, 3,..., <b>Number of entity input ports</b> .	After entity departure	1

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

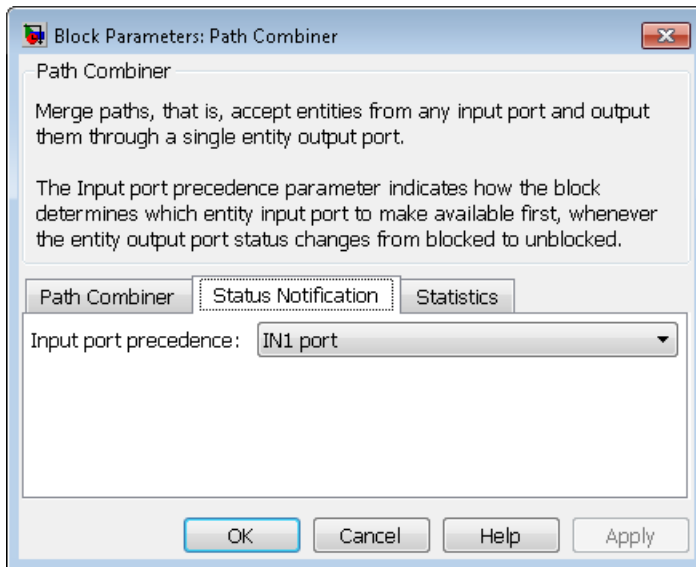
### Path Combiner Tab



#### Number of entity input ports

Determines how many entity input ports the block has.

## Status Notification Tab



### Input port precedence

Determines which entity input port the block makes available first, upon an event that changes the entity output port from blocked to unblocked.

### Initial seed

A nonnegative integer that initializes the random number generator used to select an entity input port for first notification about status changes. You see this field only if you set **Input port precedence** to Equiprobable.

### Resolve simultaneous signal updates according to event priority

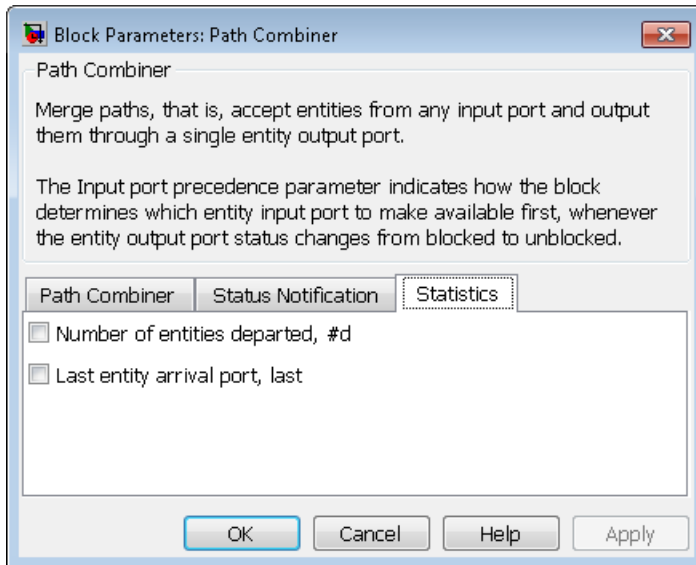
Select this option to prioritize the event that updates the port precedence explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority SYS1 on the event calendar. You see this field only if you set **Switching criterion** to From signal port p.

### Event priority

The priority of the event that updates the port precedence, relative to other simultaneous events in the simulation. **Switching criterion** to From signal port p and select **Resolve simultaneous signal updates according to event priority**.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see the preceding table, “Signal Output Ports”.



### Number of entities departed

Allows you to use the signal output port labeled **#d**.

### Last entity arrival port

Allows you to use the signal output port labeled **last**.

## See Also

Input Switch (Obsolete), Output Switch (Obsolete)

Introduced before R2006a

## Priority Queue (Obsolete)

Store entities in sorted sequence for undetermined length of time

### Library

Queues



This block stores up to  $N$  entities simultaneously in a sorted sequence, where  $N$  is the **Capacity** parameter value. The queue sorts entities according to the values of an attribute, in either ascending or descending order. Use the **Sorting attribute name** and **Sorting direction** parameters to determine the sorting behavior. The block accepts real numbers,  $Inf$ , and  $-Inf$  as valid values of the sorting attribute.

The block attempts to output an entity through the **OUT** port but retains the entity if the **OUT** port is blocked. If an entity in this block is scheduled to time out, then it might depart prematurely via the optional **TO** port. The length of time that an entity stays in this block cannot be determined in advance. The **IN** port is unavailable whenever this block stores exactly  $N$  entities. In this case, the queue is said to be full.

While you can view the value of the sorting attribute as an entity priority, this value has nothing to do with event priorities or block priorities.

### Ports

#### Entity Input Ports

Label	Description
IN	Port for arriving entities, which will be stored.

#### Entity Output Ports



Label	Description
<b>OUT</b>	Port for departing entities that do not time out while in this block.
<b>TO</b>	Port for entities that time out while in this block. You see this port only if you select <b>Enable TO port for timed-out entities</b> . This port must not be blocked when an entity attempts to depart here.

### Signal Output Ports

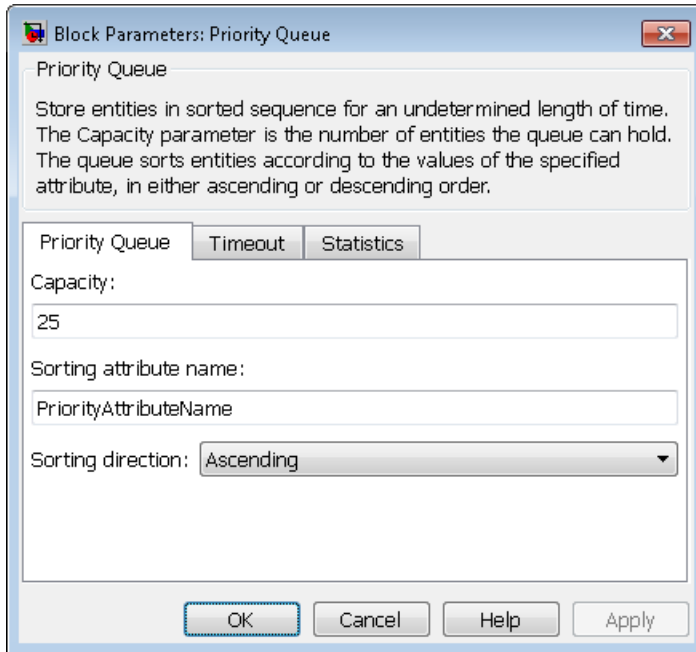
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
<b>#d</b>	Number of entities that have departed from this block via the <b>OUT</b> port since the start of the simulation.	After entity departure via the <b>OUT</b> port	3
<b>#n</b>	Number of entities currently in the queue.	After entity arrival and after entity departure	2
<b>w</b>	Sample mean of the waiting times in this block for all entities that have departed via any port.	After entity departure	1
<b>len</b>	Average number of entities in the queue over time, that is, the time average of the <b>#n</b> signal.	After entity arrival and after entity departure.	1
<b>#to</b>	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the <b>TO</b> port	3

Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

### Priority Queue Tab



#### Capacity

Determines how many entities the block can store at a time. The capacity must be a positive integer or `Inf`.

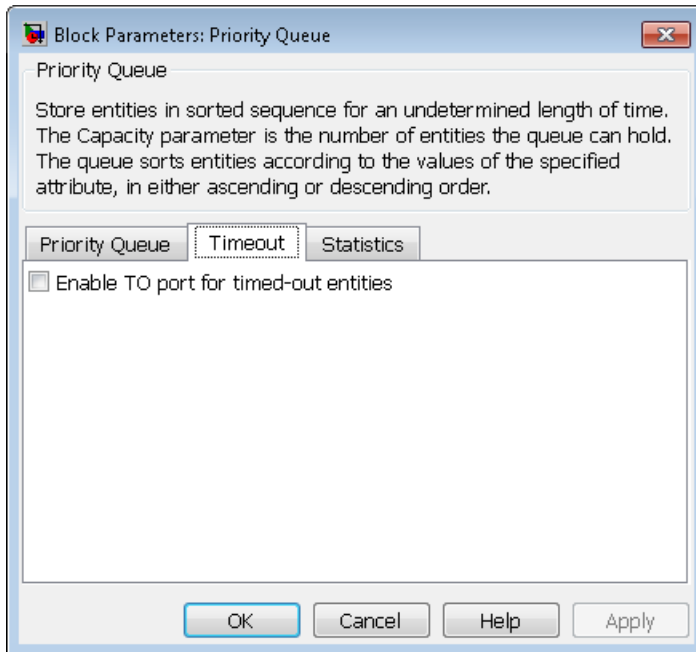
#### Sorting attribute name

The block uses this attribute to sort entities in the queue.

#### Sorting direction

Determines whether the entity at the head of the queue is the one with the smallest (**Ascending**) or largest (**Descending**) value of the attribute named above. Entities sharing the same attribute value are sorted in FIFO sequence.

## Timeout Tab

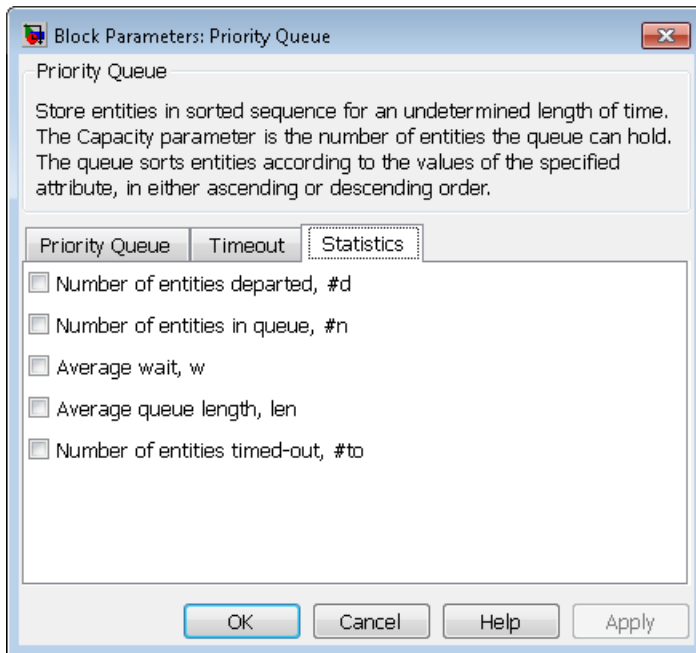


### Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the Schedule Timeout (Obsolete) block.

## Statistics Tab

These parameters determine whether certain ports produce data throughout the simulation, produce data only when you stop or pause the simulation, or are omitted from the block. For descriptions of the affected ports, see the preceding table, "Signal Output Ports".



### **Number of entities departed**

Allows you to use the signal output port labeled **#d**.

### **Number of entities in queue**

Allows you to use the signal output port labeled **#n**.

### **Average wait**

Allows you to use the signal output port labeled **w**.

### **Average queue length**

Allows you to use the signal output port labeled **len**.

### **Number of entities timed out**

Allows you to use the signal output port labeled **#to**.

## **Examples**

- “Serve Preferred Customers First”

## See Also

FIFO Queue (Obsolete), LIFO Queue (Obsolete), Single Server (Obsolete)

“Sort by Priority”

**Introduced before R2006a**

## Read Timer (Obsolete)

Report statistical data about named timer associated with arriving entities

### Library

Timing



This block reads the value of a timer that the **Start Timer (Obsolete)** block previously associated with the arriving entity. Using the **Report elapsed time** and **Report average elapsed time** parameters, you can configure the block to report the following statistics via the **et** and **w** signal output ports, respectively:

- The instantaneous value from the named timer associated with the arriving entity
- The average of **et** values among all entities that have arrived at this block during the simulation and possessed a timer of the specified name

---

**Note:** If the arriving entity does not possess a timer of that name, then you can configure the block to either produce an error or ignore the timer's absence. In the latter case, the output signals maintain their previous values.

---

The timer continues timing after the entity departs from this block, which is relevant if the same entity arrives at another **Read Timer** block later in the simulation.

### Ports

#### Entity Input Ports

Label	Description
IN	Port for arriving entities.

**Entity Output Ports**

Label	Description
OUT	Port for departing entities.

**Signal Output Ports**

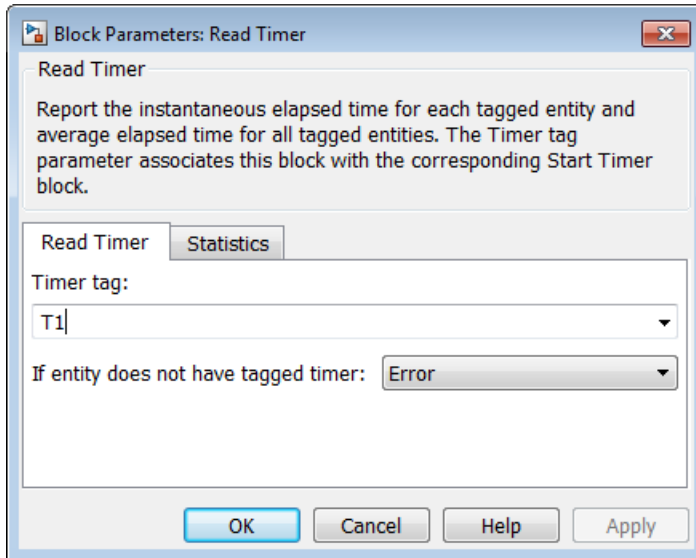
Label	Description	Time of Update When Statistic Is On	Order of Update
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	3
#t	Total number of entities that have departed from this block and possessed a timer of the specified name.	After entity departure	2
et	Instantaneous elapsed time for the arriving entity, if it possesses a timer of the specified name.	After entity departure	2
w	Average among the <b>et</b> values for all entities that have arrived at this block and possessed a timer of the specified name.	After entity departure	1

Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

### Read Timer Tab



#### Timer tag

Name of the timer to read. This timer tag corresponds to the **Timer tag** parameter of a **Start Timer** block in the model.

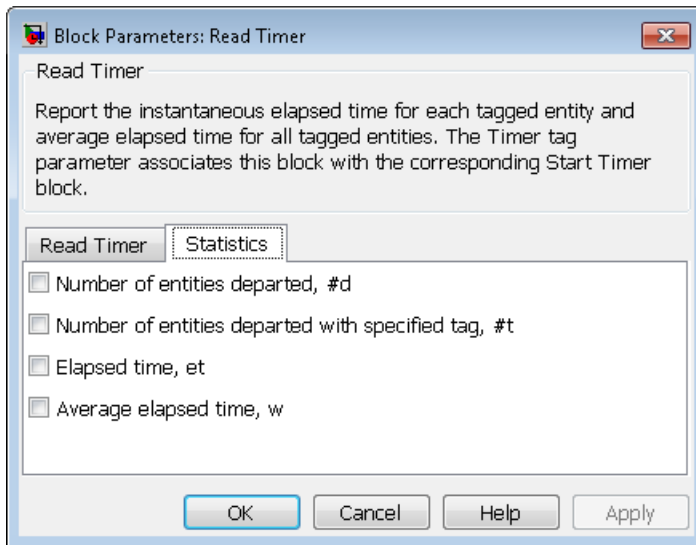
#### If entity does not have tagged timer

Behavior of the block if an arriving entity does not possess a timer with the specified timer tag.

#### Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports.





### Number of entities departed

Allows you to use the signal output port labeled **#d**.

### Number of entities departed with specified tag

Allows you to use the signal output port labeled **#t**. If you set **If entity does not have tagged timer** to **Ignore**, then the **#t** value might be less than the **#d** value.

### Elapsed time

Allows you to use the signal output port labeled **et**.

### Average elapsed time

Allows you to use the signal output port labeled **w**.

## See Also

Start Timer (Obsolete)

Introduced before R2006a

## Release Gate (Obsolete)

Allow one pending entity to arrive when event occurs

### Library

Gates



This block permits the arrival of one pending entity when a signal-based event or function call occurs; at all other times, the entity input port of the block is unavailable. By definition, the opening of the gate permits one pending entity to arrive if the entity is able to advance immediately to the next block.

No simulation time passes between the opening and subsequent closing of the gate. The gate opens and then closes in the same time instant. If no entity is already pending when the gate opens, then the gate closes without processing any entities.

The **Open gate upon** parameter determines the type of event that opens the gate:

- Sample time hits of a signal
- Edges in a trigger signal
- Changes in the numerical value of a signal
- Function calls

### Ports

#### Entity Input Ports

Label	Description
IN	Port for arriving entities.

### Signal Input Ports

Label	Description
ts	When this signal has an update, the gate opens. This signal must be an event-based signal. You see this port only if you set <b>Open gate upon</b> to <b>Sample time hit</b> from port ts.
tr	When this signal satisfies the specified trigger criteria, the gate opens. This signal must be an event-based signal. You see this port only if you set <b>Open gate upon</b> to <b>Trigger</b> from port tr.
vc	When this signal satisfies the specified value-change criteria, the gate opens. This signal must be an event-based signal. You see this port only if you set <b>Open gate upon</b> to <b>Change in signal</b> from port vc.
fcn	When this signal carries a function call, the gate opens. This signal must be an event-based function call. You see this port only if you set <b>Open gate upon</b> to <b>Function call</b> from port fcn.

### Entity Output Ports

Label	Description
OUT	Port for departing entities.

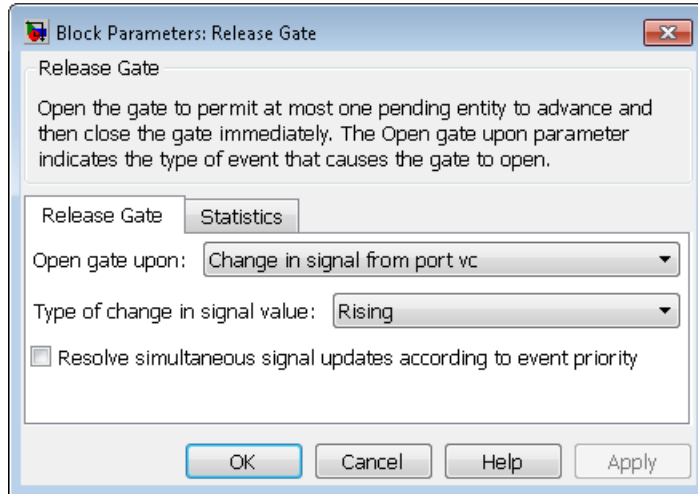
### Signal Output Ports

Label	Description	Time of Update When Statistic Is On
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure

The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

## Dialog Box

### Release Gate Tab



#### Open gate upon

Determines the type of event that causes the gate to open instantaneously.

#### Trigger type, Type of change in signal value

**Trigger type** determines whether rising, falling, or either type of trigger edge causes the gate to open. You see this field only if you set **Open gate upon** to **Trigger** from port **tr**.

**Type of change in signal value** determines whether rising, falling, or either type of value change causes the gate to open. You see this field only if you set **Open gate upon** to **Change in signal from port vc**.

#### Resolve simultaneous signal updates according to event priority

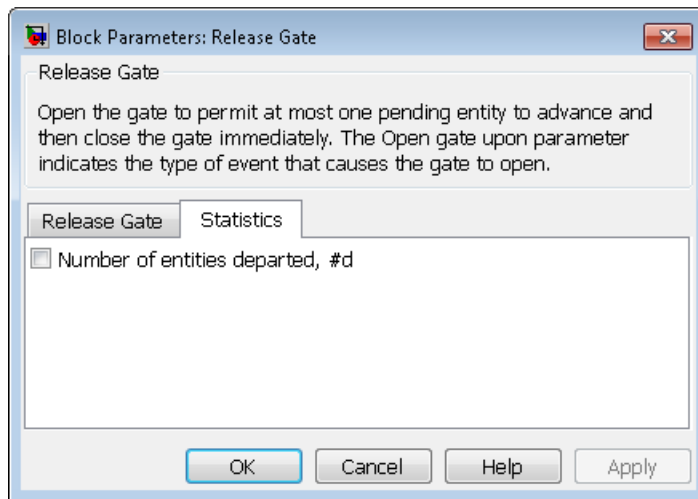
Select this option to prioritize the gate-opening event explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority **SYS1** on the event calendar.

#### Event priority

The priority of the gate-opening event, relative to other simultaneous events in the simulation. You see this field only if you select **Resolve simultaneous signal updates according to event priority**.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports.



### Number of entities departed

Allows you to use the signal output port labeled **#d**.

## See Also

Enabled Gate (Obsolete)

Introduced before R2006a

## Replicate (Obsolete)

Output copies of entity

### Library

Routing

### Description



This block outputs a copy of the arriving entity through each entity output port that is not blocked. You specify the number of copies that the block makes, using the **Number of entity output ports** parameter.

When the block replicates an entity that is subject to a timeout, all departing entities share the same expiration time; that is, the timeout events corresponding to all departing entities share the same scheduled event time. Logistically, the block cancels the timeout event of the arriving entity and schedules new timeout events for the departing entities.

### Complete or Partial Replication

The **Replicate entity when** parameter affects the circumstances under which the block accepts an entity to replicate. Choices are in the table below.

Parameter Value	Description
All entity output ports are not blocked	The block accepts an entity to replicate only when all entity output ports are connected to available ports of subsequent blocks.
Any entity output port is not blocked	The block accepts an entity to replicate when at least one entity output port is connected to an available port of a subsequent block.

If you connect multiple copies of this block, you can implement logical combinations of the parameter values in the table.

## Departure of Copies

Each time the block replicates an entity, the copies depart in a sequence whose start is determined by the **Departure port precedence** parameter. Choices are in the table below.

Parameter Value	Description	Example
OUT1 port	Each time the block replicates an entity, the copies depart via entity output ports <b>OUT1</b> , <b>OUT2</b> , <b>OUT3</b> ,..., in that sequence.	The sequence of departures is always <b>OUT1</b> , <b>OUT2</b> , <b>OUT3</b> ,... throughout the simulation.
Round robin	Each time the block replicates an entity, the first copy departs via the port after the one that received preference on the last such occasion. The remaining copies depart via the subsequent ports in turn.	On a block with three entity output ports, the first time the block replicates an entity, the copies depart in the sequence <b>OUT1</b> , <b>OUT2</b> , <b>OUT3</b> . The second time, the copies depart in the sequence <b>OUT2</b> , <b>OUT3</b> , <b>OUT1</b> . The third time, the copies depart in the sequence <b>OUT3</b> , <b>OUT1</b> , <b>OUT2</b> . The fourth time is analogous to the first time, and so on.
Equiprobable	Each time the block replicates an entity, the first copy departs via a randomly selected entity output port. All entity output ports are equally likely to be selected and the <b>Initial seed</b> parameter initializes the random number generation process. The remaining copies depart via the subsequent ports in turn.	On a block with four entity output ports, if the random number is three, then the copies depart in the sequence <b>OUT3</b> , <b>OUT4</b> , <b>OUT1</b> , <b>OUT2</b> . If the random number is two on the next such occasion, then the copies depart in the sequence <b>OUT2</b> , <b>OUT3</b> , <b>OUT4</b> , <b>OUT1</b> .

An example in which the choice of **Departure port precedence** parameter is relevant is a model that sets an attribute on each replicated entity based on its departure port

and then advances all replicated entities along a merged path to a **FIFO Queue** block. At each replication occurrence during the simulation, the **Departure port precedence** parameter determines the sequence of the replicated entities in the queue.

In some cases, a departure through one entity output port causes another entity output port to become newly blocked. For example, this could happen if two entity output ports connect to a **Path Combiner** block, which in turn connects to a **Single Server** block whose service time is nonzero. Use the **If an output port becomes blocked during replication** parameter to determine how the block responds. Choices are in the table below.

Parameter Value	Description
Discard entity	The block discards the entity that is supposed to depart through the newly blocked entity output port.
Warn and discard entity	The block issues a warning message in the MATLAB Command Window, and discards the entity that is supposed to depart through the newly blocked entity output port.
Error	The simulation halts with an error message.

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities.

### Entity Output Ports

Label	Description
OUT1, OUT2, OUT3, and so on	Port for departing entities, which are copies of the arriving entity. The <b>Number of entity output ports</b> parameter determines how many of these entity input ports the block has.



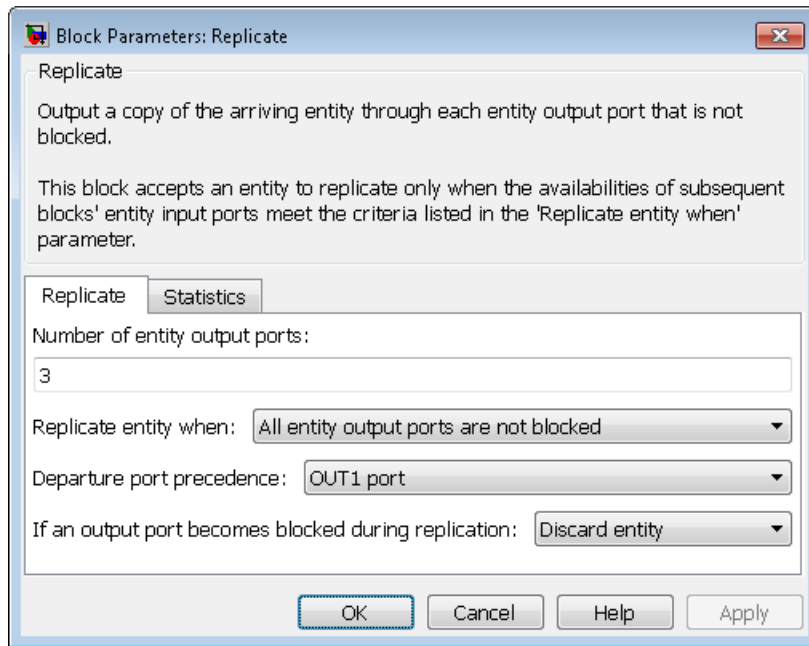
**Signal Output Ports**

<b>Label</b>	<b>Description</b>	<b>Time of Update When Statistic Is On</b>	<b>Order of Update</b>
<b>#a</b>	Number of entities that have arrived at this block since the start of the simulation.	After entity arrival	1
<b>#d</b>	Number of entities that have departed from this block since the start of the simulation.	After each entity departure	2

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

### Replicate Tab



#### Number of entity output ports

Determines how many entity output ports the block has; that is, the maximum number of copies the block makes for each arriving entity.

#### Replicate entity when

Determines whether the block is available to arriving entities whenever at least one entity output port is not blocked, or only when all entity output ports are not blocked.

#### Departure port precedence

Determines the start of the sequence in which the block outputs the copies, each time the block replicates an entity.

#### Initial seed

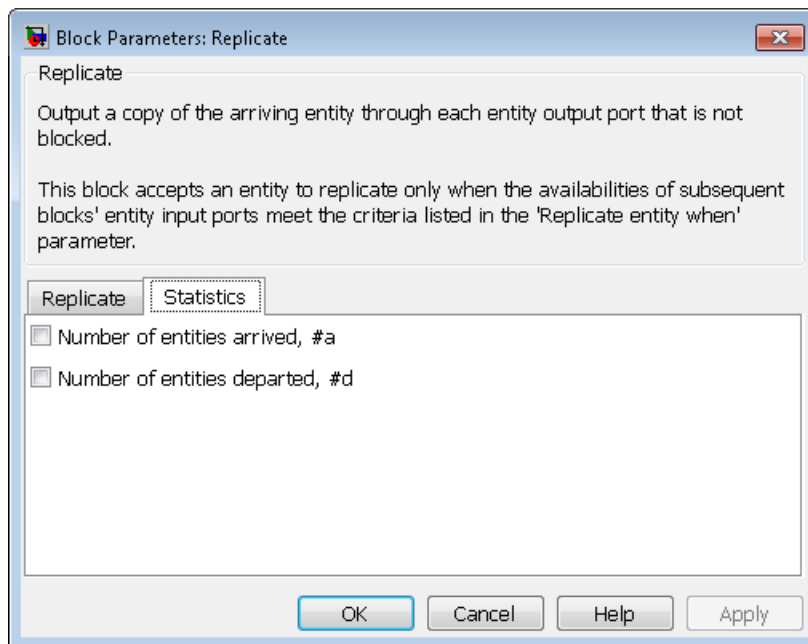
A nonnegative integer that initializes the random number generator used to determine the output sequence. You see this field only if you set **Departure port precedence** to Equiprobable.

### If an output port becomes blocked during replication

Determines whether the block issues a message when a replicated entity is unable to depart because an output port becomes blocked during the replication process. You see this field only if you set **Replicate entity when** to All entity output ports are not blocked.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see the preceding table, “Signal Output Ports”.



### Number of entities arrived

Allows you to use the signal output port labeled **#a**.

**Number of entities departed**

Allows you to use the signal output port labeled **#d**.

**See Also**

Event-Based Entity Generator (Obsolete), Path Combiner (Obsolete)

“Replicate Entities on Multiple Paths”

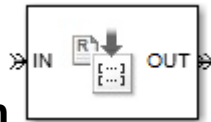
**Introduced before R2006a**

# Resource Acquire (Obsolete)

Acquire resource

## Library

Entity Management



## Description

This block accepts an entity that requests the use of resources, assigns resources to it, and then outputs it.

You can specify the resource types and amounts for the entity. The block stores the assigned resource with the entity, where each resource has a name and a value.

You can optionally specify a timeout that limits the maximum duration an entity waits for resources. You can also prioritize how resources are granted.

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities.

### Entity Output Ports

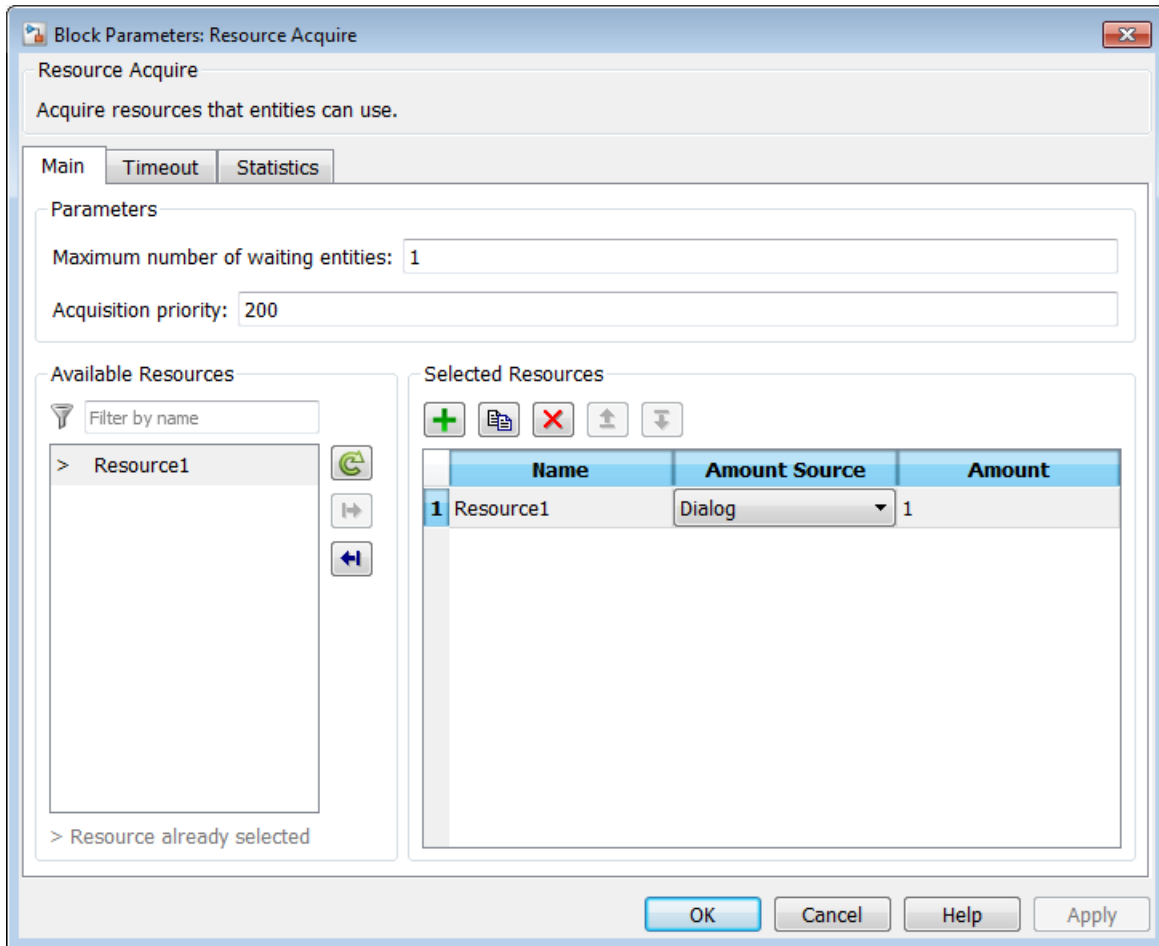
Label	Description
OUT	Port for departing entities that have acquired resources.

### Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block since the start of the simulation or since the last reset.	After entity departure	3
#n	Number of entities currently in the block, between 0 and $N$ .	After entity arrival and after entity departure	2
w	Sample mean of the waiting times in this block for all entities that have departed via any port. If the OUT port is blocked when the entity completes service, an entity waiting time exceeds its service time.	After entity departure	1
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the TO port	3

# Dialog Box

## Main Tab



### Maximum number of waiting entities

Enter maximum number of entities that can wait for a resource.

### Acquisition priority

Enter priority number for resource acquisition. This number prioritizes the Resource Acquire block that has higher priority in a model that has multiple Resource Acquire blocks. A smaller numeric value indicates higher priority.

### Available Resources




Use the **Available Resources** controls to.

- Select the resources from the resources defined in all the **Resource Pool** blocks in the model.
- Add the resources to the Selected Resources table, where you can configure resource acquisition details.

The list displays all the available resources in the model. (If there are no resources, the **Available Attributes** list is empty.)

If the resource list is long, you can type the resource name in the text box to filter the list.

Use the buttons in the **Available Resources** section to help build the resources table. The buttons perform these actions:

Button	Action
	Refresh the <b>Available Resources</b> list. The list updates with any upstream model changes you make while the block dialog box is open.
	Add the selected resources to the <b>Selected Resources</b> table.
	Move the selected resource from the <b>Selected Resources</b> table to the <b>Available Resources</b> list.  Note: If the selected resource is one you added manually, this button appears dimmed.

The message area below the available resources list displays additional messages about the resources, as they apply.

Message	Meaning
> Resource already selected	You have already added the resource to the <b>Selected Resources</b> table. You cannot add the resource to the table again.








## Selected Resources

Use the controls under **Selected Resources** to build and manage the list of resources to attach to the entity. Each resource appears as a row in a table.

Using these controls, you can:

- Add a resource manually.
- Modify a resource that you added to the table from the **Available Resources** list to attach to the entity.

The buttons under **Selected Resources** perform these actions:

Button	Action	Notes
	Add a template resource to the table.	Rename the resource and specify its properties.
	Add a copy of the selected resource to the table to use as the basis of a new resource.	Rename the copy. Two resources cannot have the same name.
	Remove the selected resource from the <b>Selected Resources</b> table.	When you delete a resource this way, no confirmation appears and you cannot undo the operation.
	Move the selected resource up in order in the <b>Selected Resources</b> table.	NA
	Move the selected resource down in order in the <b>Selected Resources</b> table.	NA

**Note:** If you delete a row and apply the change, the deletion can affect signal output ports corresponding to other attributes. For example, if the block has a signal output port **A2** and you delete the attribute with a port marked **A1**, the block relabels **A2** as **A1**. Verify that any signal that connects to the relabeled port is still connected as you expect.

Property	Specify	Use
<b>Name</b>	The name of the resource. Each resource must have a unique name.	Double-click the existing name, and then type the new name.
<b>Amount Source</b>	Whether the resource amount, that an entity requests, comes from the dialog box or an attribute.	Select <b>Dialog</b> or <b>Attribute</b> . If you select <b>Attribute</b> , the source of the resource amount comes from the attribute of the entity. This option allows each entity to acquire varying amounts of resources. For more information, see “Set Resource Amount with Attributes”
<b>Amount</b>	The value to assign to the resource (when the resource comes from the dialog box).	Double-click the value, and then type the value you want to assign.  This value is the number of resources acquired per entity. For example, if <b>Amount</b> is <b>3</b> , each entity that arrives at the <b>Release Acquire</b> must wait to acquire 3 resources before departing the block.

## Timeout Tab

### Enable TO port for timed-out entities

Select to use the signal output port labeled **TO**.

## Statistics Tab

### Number of entities departed, #d

Select to use the signal output port labeled **#d**.

**Number of entities in block, #n**

Select to use the signal output port labeled #n.

**Average wait, w**

Select to use the signal output port labeled w.

**Number of entities timed out, #to**

Select to use the signal output port labeled #to.

## See Also

Resource Pool (Obsolete), Resource Release (Obsolete)

“Model with Resources”

**Introduced in R2015a**

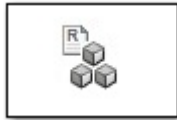
## Resource Pool (Obsolete)

Define resource

## Library

Entity Management

### Description



This block defines resources that entities can use during model simulation. Use the **Resource Acquire** and **Resource Release** blocks to work with these resources.

Initialize the block with specified amount of available resources. Then:

- Use one or more **Resource Acquire** blocks to reserve the use of those resources.
- Use a **Resource Release** block to return resources back to this block for future use.

Resources are visible to the current subsystem and its children. Resource are not visible to parent subsystems.

## Ports

### Resource Input Ports

Label	Description
<b>amount</b>	Port for arriving signals to specify the amount of the resource. You see this port only if you set <b>Resource amount source</b> to <b>Signal port</b> .

### Signal Output Ports

Label	Description	Time of Update When Statistic is On
<b>#u</b>	Port to display the number of resources in use. You see this port only if you set	1

Label	Description	Time of Update When Statistic is On
	Resource amount source to Signal port.	
util	Port to display the average use of the resources. You see this port only if you select the <b>Average utilization, util</b> check box.	2

## Dialog Box

### Main Tab

Block Parameters: Resource Pool

Resource Pool

Define a resource that entities can acquire, use, and release.

Main Statistics

Resource name:  
Resource1

Resource granularity: Discrete unit

Reusable upon release

Resource amount source: Dialog

Resource amount:  
10

OK Cancel Help Apply

#### Resource name

Enter name of entity resource.

#### Resource granularity

Select granularity of resource use.

- **Discrete unit** — Use whole number increment.
- **Fractional amount** — Use fractional increment.

### **Reusable upon release**

- Select this check box to allow this resource to return to the resource pool upon release. An example of such a resource is a table in a restaurant, which is available for reuse when a customer leaves.

Selecting this check box enables the **Resource amount source** check box.

- Clear this check box to prevent this resource from returning to the resource pool upon release. An example of such a resource is food in a restaurant, which is not reusable upon consumption.

### **Resource amount source**

Select resource amount source.

- **Dialog**

Selecting this option enables the **Resource amount** parameter.

- **Signal port**

The block derives the resource amount from the signal port.

### **Resource amount**

Enter amount of resource.

## **Statistics Tab**

### **Amount in use, #u**

Select to use the signal output port labeled **#u**.

### **Average utilization, util**

Select to use the signal output port labeled **util**.

## **See Also**

Resource Acquire (Obsolete), Resource Release (Obsolete)

“Model with Resources”

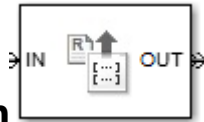
**Introduced in R2015a**

## Resource Release (Obsolete)

Release resources that entities do not need

### Library

Entity Management



### Description

This block releases the use of resources for a passing entity. You can specify that the block release certain resource types or release all resources.

### Ports

#### Entity Input Ports

Label	Description
IN	Port for arriving entities.

#### Entity Output Ports

Label	Description
OUT	Port for departing entities that have released use of their resources.

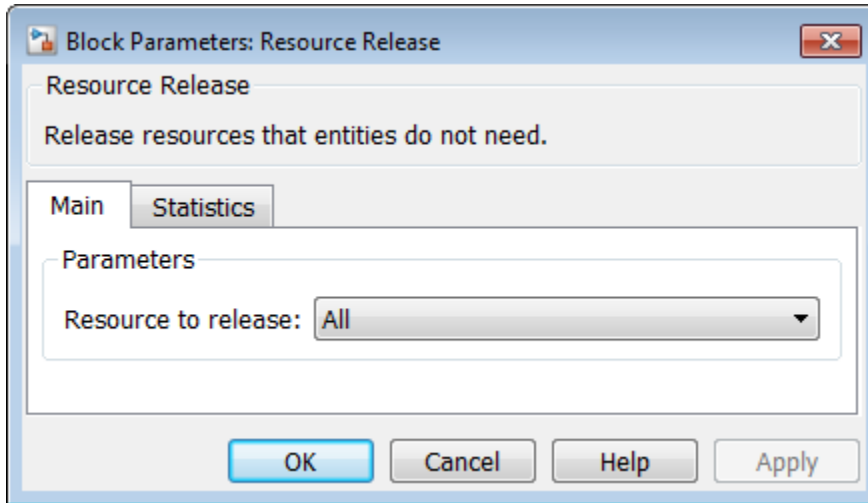
#### Signal Output Ports

Label	Description	Time of Update When Statistic Is On
#d	Number of entities that have departed from this block since the start of the simulation or since the last reset.	After entity departure



## Dialog Box

### Main Tab



### Resource to Release

Select the resources to release.

- All

Release the use of all resources for a passing entity.

- Selected

Release selected resources. Selecting this option enables the **Available Resources** table.

### Available Resources

Use the **Available Resources** controls to:




- Select the resources from the resources defined in all the Resource Pool blocks in the model

- Add the resources to the Selected Resources table, where you can modify them.

The list displays all the resources in the model. (If there are no resources, the **Available Resources** list is empty).

If the resource list is long, you can type the resource name in the text box to filter the list.

Use the buttons in the **Available Resources** section to help build the resources table. The buttons perform these actions:

Button	Action
	Refresh the <b>Available Resources</b> list. The list updates with any upstream model changes you make while the block dialog box is open.
	Add the selected resources to the <b>Selected Resources</b> table.
	Move the selected resource from the <b>Selected Resources</b> table to the <b>Available Resources</b> list.  Note: If the selected resource is one you added manually, this button appears dimmed.

The message area below the available resources list displays additional messages about the resources, as they apply.

Message	Meaning
> Resource already selected	You have already added the resource to the <b>Selected Resources</b> table. You cannot add the resource to the table again.

### Selected Resources






Use the controls under **Selected Resources** to build and manage the list of resources to release. Each resource appears as a row in a table.

Using these controls, you can:

- Add a resource manually.

- Modify a resource that you added to the table from the **Available Resources** list to release.

The buttons under **Selected Resources** perform these actions:

Button	Action	Notes
	Add a template resource to the table.	Rename the resource and specify its properties.
	Add a copy of the selected resource to the table to use as the basis of a new resource.	Rename the copy. Two resources cannot have the same name.
	Remove the selected resource from the <b>Selected Resources</b> table.	When you delete a resource this way, no confirmation appears and you cannot undo the operation.
	Move the selected resource up in order in the <b>Selected Resources</b> table.	N/A
	Move the selected resource down in order in the <b>Selected Resources</b> table.	N/A

**Note:** If you delete a row and apply the change, the deletion can affect signal output ports corresponding to other attributes. For example, if the block has a signal output port **A2** and you delete the attribute with a port marked **A1**, the block relabels **A2** as **A1**. Verify that any signal that connects to the relabeled port is still connected as you expect.

Property	Specify	Use
Name	The name of the resource to release.	Double-click the existing name, and then type the new name.

## Statistics Tab

Number of entities departed, #d

Select to use the signal output port labeled **#d**.

## **See Also**

Resource Acquire (Obsolete), Resource Pool (Obsolete)

“Model with Resources”

**Introduced in R2015a**

# Schedule Timeout (Obsolete)

Schedule timeout event for each entity

## Library

Timing



## Description

This block schedules a timeout event for each arriving entity. Timeout events enable you to limit the time that an entity spends on designated entity paths during the simulation. Topologically, this block designates a beginning of an entity path that is relevant to the time limit.

## Characteristics of Timeout Event

The timeout event is on the event calendar and has these characteristics:

- Event time equal to the entity's arrival time plus a timeout interval. You specify the timeout interval via a parameter, attribute, or signal, depending on the **Timeout interval from** parameter value. The block determines the absolute event time of an entity's timeout event upon the entity's arrival.

---

**Note:** If you specify the timeout interval via an event-based signal, be sure that its updates occur before the entity arrives.

---

For example, if an entity arrives at  $T=5$  and the timeout interval is 3 (seconds), then the block schedules the timeout event to occur at  $T=5+3=8$ .

- A name that you specify via the **Timeout tag** parameter. The event calendar can contain multiple independent timeout events for the same entity, as long as they have

distinct timeout tags. This block does not affect timeout events having other timeout tags.

- Event priority that you specify via the **Timeout event priority** parameter. Note that if timeout events for two entities have distinct event priorities and are scheduled for the same value, or sufficiently close values, of the simulation clock, then the priority values determine which entity times out first.

### Occurrence of Timeout Event

If the timeout event occurs for a specific entity, then that entity attempts to depart from a **TO** entity output port of the storage block in which it resides. To configure a block so that it has a **TO** port, select the **Enable TO port for timed-out entities** parameter in the block's dialog box. If the timeout event occurs while the entity is in a block that has no **TO** port, then the **Schedule Timeout** block's **If entity has no destination when timeout occurs** parameter indicates whether the simulation halts with an error message, or discards the entity while issuing a warning.

To cancel a timeout event before it occurs, use the **Cancel Timeout (Obsolete)** block. You cannot directly change the scheduled time or priority of a timeout event that is already on the event calendar. You can, however, cancel a timeout event and subsequently schedule a new one having the same timeout tag.

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities.

### Signal Input Ports

Label	Description
ti	Timeout interval for a newly arrived entity. This signal must be an event-based signal. You see this port only if you set <b>Timeout interval from</b> to <b>Signal port ti</b> .

### Entity Output Ports

Label	Description
OUT	Port for entities whose timeout event the block has just scheduled.

### Signal Output Ports

Label	Description	Time of Update When Statistic Is On
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure

The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

## Dialog Box

### Schedule Timeout Tab

Block Parameters: Schedule Timeout

Schedule Timeout

Define the beginning of a timeout region by scheduling a timeout event for each passing entity.

Schedule Timeout Statistics

Timeout tag:  
TO1

Timeout interval from: Dialog

Timeout interval:  
1

Timeout event priority:  
1700

If timeout is already scheduled: Error

If entity has no destination when timeout occurs: Error

OK Cancel Help Apply

#### Timeout tag

Name of the timeout to associate with each entity. Enter a new timeout tag, or reschedule a previous timeout by choosing it in the drop-down list.

#### Timeout interval from

Determines whether the timeout interval is computed from a parameter in this dialog box, an input signal, or an attribute of the arriving entity.

#### Timeout interval

The length of time between an entity's arrival time and the scheduled timeout event for that entity. You see this field only if you set **Timeout interval from** to Dialog.



**Attribute name**

The name of the attribute whose value the block uses as the timeout interval for an entity. You see this field only if you set **Timeout interval from** to **Attribute**.

**Timeout event priority**

The priority of the timeout event, relative to other simultaneous events in the simulation.

**If timeout is already scheduled**

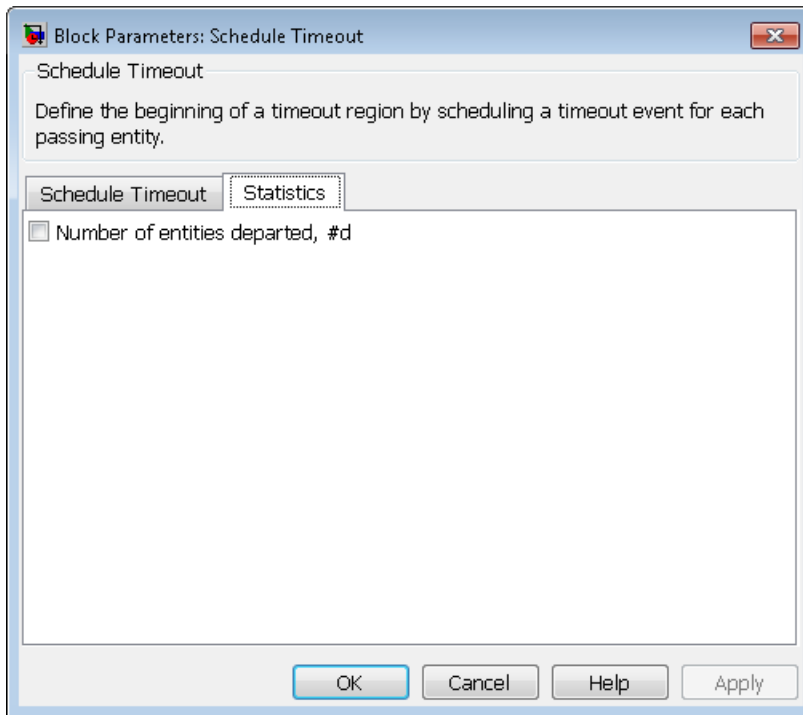
Behavior of the block if a timeout event with the specified timeout tag is already scheduled for the arriving entity.

**If entity has no destination when timeout occurs**

Behavior of the block if a timeout event occurs for an entity that resides in a block that has no visible **TO** entity output port.

**Statistics Tab**

These parameters determine whether the block produces data at signal output ports or omits those ports.



### **Number of entities departed**

Allows you to use the signal output port labeled **#d**.

## **See Also**

Cancel Timeout (Obsolete)

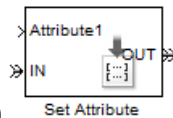
**Introduced in R2007a**

# Set Attribute (Obsolete)

Assign data to entity

## Library

Attributes

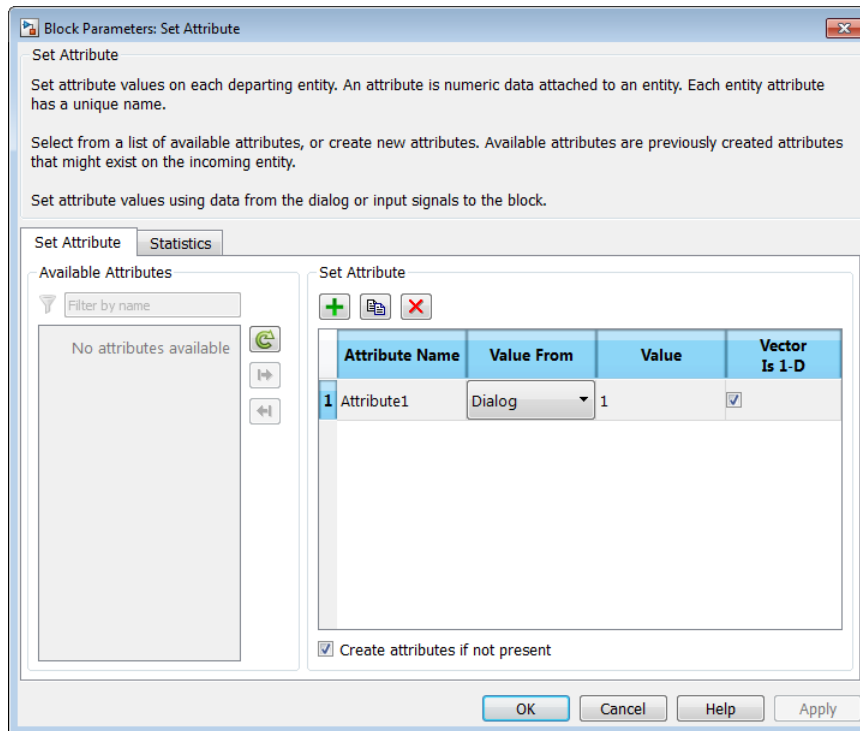


## Description

This block accepts an entity, assigns data to it, and then outputs it. Assigned data is stored in entity attributes. Each attribute has a name and a value that you specify. You can specify up to 32 attributes in the block. To learn about the kind of data an attribute can store, see “Attribute Value Support”.

## Dialog Box

### Set Attribute Tab



#### Available Attributes




Use the **Available Attributes** controls to:

- Select the attributes from incoming entity paths that you want to access on the departing entity.
- Add the attributes to the **Set Attribute** table, where you can modify them.

The list displays all the attributes on all the incoming entities. (If the entity paths entering the Set Attribute block do not have any attributes, the **Available Attributes** list is empty).

If the attribute list is long, you can type the attribute name in the text box to filter the list.

Use the buttons in the **Available Attributes** section to help build the attributes table. The buttons perform these actions:

Button	Action
	Refresh the <b>Available Attributes</b> list. This action updates the list with any upstream model changes you make while the block dialog box is open.
	Add the selected attribute to the <b>Set Attribute</b> table.
	Move the selected attribute from the <b>Set Attribute</b> table to the <b>Available Attributes</b> list.  Note: If the selected attribute is one you added manually, this button appears dimmed.

The message area below the available attributes list displays additional messages about the attributes, as they apply.

Message	Meaning
> Attribute already selected	You have already added the attribute to the <b>Set Attribute</b> table. You cannot add the attribute to the table again.
* Attribute may not be present	When multiple entity paths enter the block, all entities might not have the same attributes. Attributes that are not on all entering entities display an asterisk in the list, and this message appears. If you add such an attribute to the <b>Set Attribute</b> table, the behavior depends on how the <b>Create attribute if not present</b> check box is set.

**Create attribute if not present**

Check box that enables the block to define new attributes when an attribute in the table is not present in the current entity. If the check box is deselected, the simulation issues an error if an attribute named in the table does not already exist.

Select the check box if you want to:

- Set an attribute on each departing entity that previously existed on only certain incoming entity paths.
- Set a new attribute that you manually defined in the table on each departing entity.

Clear the check box if you want to:

- Protect against scenarios in which you add an attribute to the table from the **Available Attributes** list and later rename the attribute. When the check box is not selected, the renamed attribute causes an error because it no longer matches one present in the current entity.




### Set Attribute

Use the controls under **Set Attribute** to build and manage the list of attributes to attach to each departing entity. Each attribute appears as a row in a table.

Using these controls, you can:

- Add an attribute manually to attach to the entity.
- Modify an attribute that you added to the table from the **Available Attributes** list to attach to the entity.

The buttons under **Set Attribute** perform these actions:

Button	Action	Notes
	Add a template attribute to the table.	Rename the attribute and specify its properties.
	Add a copy of the selected attribute to the table to use as the basis of a new attribute.	Rename the copy. Two attributes cannot have the same name.
	Remove the selected attribute from the <b>Set Attribute</b> table.	When you delete an attribute this way, no confirmation

Button	Action	Notes
		appears and you cannot undo the operation.

**Note:** If you delete a row and apply the change, the deletion can affect signal output ports corresponding to other attributes. For example, if the block has a signal output port **A2** and you delete the attribute with a port marked **A1**, the block relabels **A2** as **A1**. Verify that any signal that connects to the relabeled port is still connected as you expect.

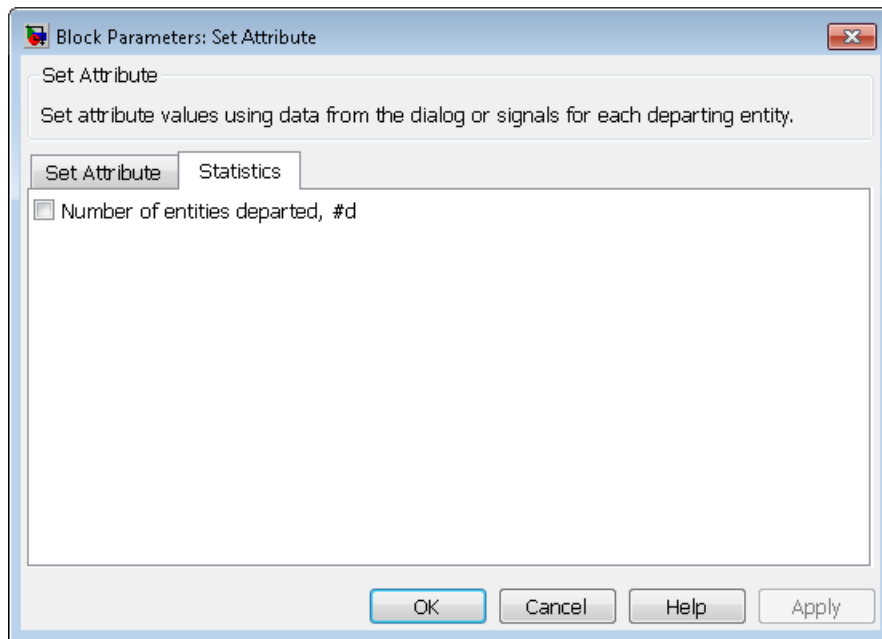
The table displays the attributes you added from the **Available Attributes** list or added manually. Use it to set these four attribute properties:

Property	Specify	Use
<b>Attribute Name</b>	The name of the attribute. Each attribute must have a unique name.	Double-click the existing name, and then type the new name.
<b>Value From</b>	Whether the data for the attribute value comes from the dialog box or a signal.	Select <b>Dialog</b> or <b>Signal port</b> . If you select <b>Signal port</b> , an input port with the name specified is added to the block after you apply your changes. When you connect a signal to the input port, the block assigns the value of the signal to the attribute during simulation.
<b>Value</b>	The value to assign to the attribute (when the attribute comes from the dialog box).	Double-click the value, and then type the value you want to assign.
<b>Vector Is 1-D</b>	Whether the block assigns the attribute as a vector of length N or as a multidimensional array when the <b>Value</b> evaluates to an N-element	Select the check box to assign the attribute as a vector of length N. Clear it to assign the attribute as a multidimensional array.

Property	Specify	Use
	row or column vector. This property applies only to values that come from the dialog box when the value evaluates to an N-element row or column vector.	

### Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see [Signal Output Ports](#).



#### Number of entities departed

Allows you to use the signal output port labeled **#d**.



## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities

### Signal Input Ports

Label	Description
<i>Attribute name</i>	Data to assign to the attribute specified in each row of the table. The signal must be a fixed-size, event-based signal. You see this port only if you set <b>Value From</b> to <b>Signal</b> port. The default <b>Name</b> that corresponds to each row is <b>Attributex</b> , where <b>x</b> = 1, 2, 3, etc.

### Entity Output Ports

Label	Description
OUT	Port for departing entities, with data assigned to them.

### Signal Output Ports

Label	Description	Time of Update When Statistic Is On
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure

The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

## Examples

- “Set Attributes”

## See Also

Get Attribute (Obsolete)

“Working with Entity Attributes”

**Introduced before R2006a**

# Signal Latch (Obsolete)

Write input signal value to memory and read memory to output signal upon events

## Library

Signal Management



## Description

The **Signal Latch** block is a versatile block for manipulating event-based signals. You can use it to delay or resample signals based on events, not time. This block stores and outputs the values of the **in** input signal based on events:

- The block writes the value of the **in** signal to an internal memory location when a “write to memory” event occurs. The **Write to memory upon** parameter indicates the type of signal-based event or function call that causes a write event.
- The block reads the memory value and updates the signal at the **out** port, if present, when a “read from memory” event occurs. The **Read from memory upon** parameter indicates the type of internal or external event that causes a read event:
  - If you set **Read from memory upon** to **Write to memory event**, then every write event causes a read event. The **out** signal is like a resampled version of the **in** signal.
  - Otherwise, the **Read from memory upon** parameter indicates the type of signal-based event or function call that causes a read event. In this case, write and read events occur independently and are not required to alternate. The **out** signal is like a delayed resampled version of the **in** signal.

This block is useful for modeling feedback loops in discrete-event systems in which an output from one component is an input to another component. Because the two components work separately in such a system, the updates of the input and output signals are independent in both causality and timing. This block lets you control the causality and timing associated with storing the output from one component and updating the value received by the other component.

## Ports

### Signal Input Ports

Label	Description
<b>wts</b>	Signal whose updates cause write events. This signal must be an event-based signal. You see this port only if you set <b>Write to memory upon</b> to <b>Sample time hit</b> from port <b>wts</b> .
<b>wtr</b>	Trigger signal whose edges cause write events. This signal must be an event-based signal. You see this port only if you set <b>Write to memory upon</b> to <b>Trigger</b> from port <b>wtr</b> .
<b>wvc</b>	Signal whose numerical changes in value cause write events. This signal must be an event-based signal. You see this port only if you set <b>Write to memory upon</b> to <b>Change in signal</b> from port <b>wvc</b> .
<b>wfcn</b>	Function-call signal that causes write events. This signal must be an event-based function call. You see this port only if you set <b>Write to memory upon</b> to <b>Function call</b> from port <b>wfcn</b> .
<b>rts</b>	Signal whose updates cause read events. This signal must be an event-based signal. You see this port only if you set <b>Read from memory upon</b> to <b>Sample time hit</b> from port <b>rts</b> .
<b>rtr</b>	Trigger signal whose edges cause read events. This signal must be an event-based signal. You see this port only if you set <b>Read from memory upon</b> to <b>Trigger</b> from port <b>rtr</b> .
<b>rvc</b>	Signal whose numerical changes in value cause read events. This signal must be an event-based signal. You see this port only if you set <b>Read from memory upon</b> to <b>Change in signal</b> from port <b>rvc</b> .
<b>rfcn</b>	Function-call signal that causes read events. This signal must be an event-based function call. You see this port only if you set <b>Read from memory upon</b> to <b>Function call</b> from port <b>rfcn</b> .
<b>in</b>	Signal to be resampled and/or delayed. This signal must be an event-based signal.

### Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update	Initial Value
<b>st</b>	0 or 1, depending on whether the block more	Upon write events and upon read events	1	0

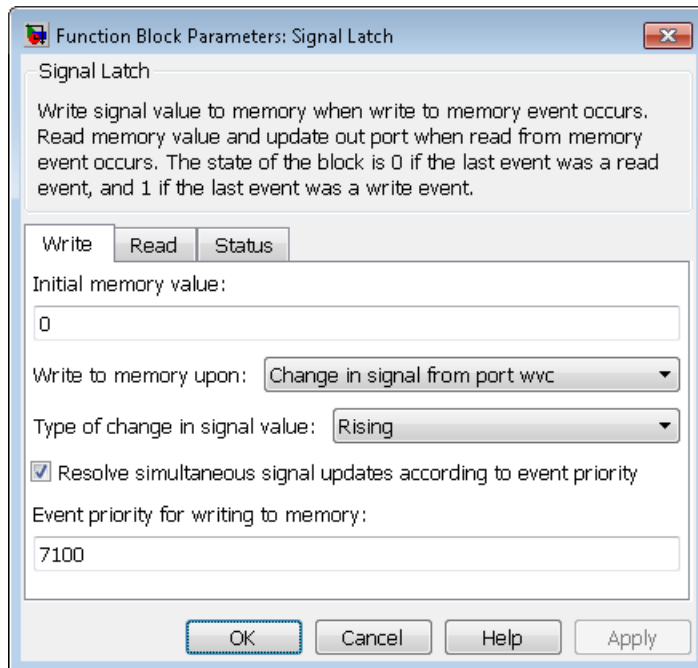
Label	Description	Time of Update When Statistic Is On	Order of Update	Initial Value
	recently processed a read or write event.			
<b>mem</b>	The value of the block's internal memory when a write event occurs.	Upon write events	1	Value of <b>Initial memory value</b> parameter
<b>out</b>	The value of the block's internal memory when a read event occurs.	Upon read events	1	

Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

The initial value is in effect from the start of the simulation until the first update by the block.

## Dialog Box

### Write Tab



#### Initial memory value

The value in the block's internal memory before the first write event occurs.

#### Write to memory upon

The type of signal-based event or function call that causes a write event.

#### Trigger type, Type of change in signal value

**Trigger type** determines whether rising, falling, or either type of trigger edge causes a write event. You see this field only if you set **Write to memory upon** to **Trigger** from port wtr.

**Type of change in signal value** determines whether rising, falling, or either type of value change causes a write event. You see this field only if you set **Write to memory upon** to **Change in signal from port wvc**.

**Resolve simultaneous signal updates according to event priority**

Select this option to control the sequencing of the write event, relative to other simultaneous events in the simulation. If you do not select this option, the application executes the write event immediately upon detecting the signal-based event that causes it.

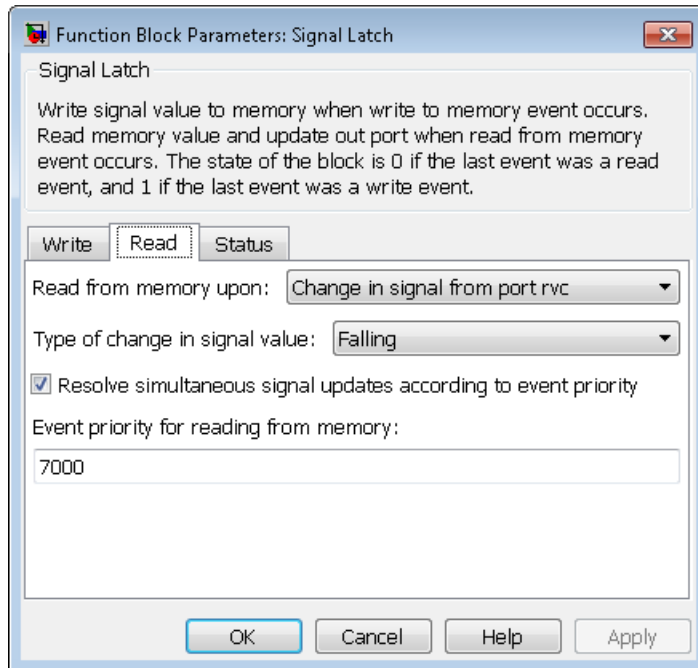
**Event priority for writing to memory**

The priority of the write event, relative to other simultaneous events in the simulation.

Use of this parameter depends on the following:

- You see this field only if you select **Resolve simultaneous signal updates according to event priority** on this tab.
- If you select **Resolve simultaneous signal updates according to event priority** on both the write tab and the read tab, the software ignores **Event priority for reading from memory**. Instead, the simulation resolves simultaneous signal updates based on only the **Event priority for writing to memory** parameter. In this case, the software executes the write event before the read event.
- If you select the configuration parameter **Prevent duplicate events on multiport blocks and branched signals** in your model, the software uses the **Event priority for writing to memory** parameter to help Simulink to sort blocks in the model. In this case, the software no longer schedules an event that you can view on the SimEvents event calendar.

## Read Tab



### Read from memory upon

The type of signal-based event, function call, or internal write event that causes a read event.

### Trigger type, Type of change in signal value

**Trigger type** determines whether rising, falling, or either type of trigger edge causes a read event. You see this field only if you set **Read from memory upon** to Trigger from port rtr.

**Type of change in signal value** determines whether rising, falling, or either type of value change causes a read event. You see this field only if you set **Read from memory upon** to Change in signal from port rvc.

### Resolve simultaneous signal updates according to event priority

Select this option to control the sequencing of the read event, relative to other simultaneous events in the simulation. If you do not select this option, the



application executes the read event immediately upon detecting the signal-based event that causes it. You see this field only if you set **Read from memory upon** to an option other than Write to memory event.

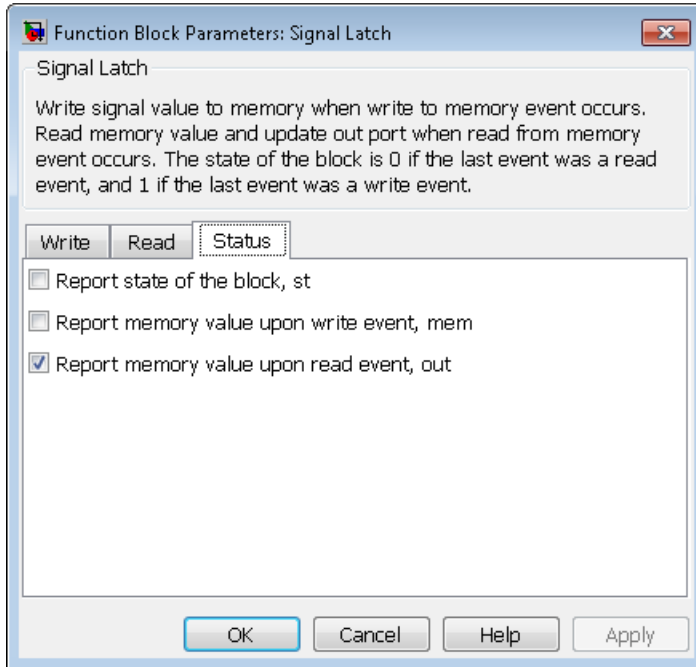
### **Event priority for reading from memory**

The priority of the read event, relative to other simultaneous events in the simulation.

Use of this parameter depends on the following:

- You see this field only if you select **Resolve simultaneous signal updates according to event priority** on this tab.
- If you select **Resolve simultaneous signal updates according to event priority** on both the write tab and the read tab, the software ignores **Event priority for reading from memory**. Instead, the simulation resolves simultaneous signal updates based on only the **Event priority for writing to memory** parameter. In this case, the software executes the write event before the read event.
- If you select the configuration parameter **Prevent duplicate events on multiport blocks and branched signals** in your model, the software uses the **Event priority for writing to memory** parameter to help Simulink to sort blocks in the model. In this case, the software no longer schedules an event that you can view on the SimEvents event calendar.

## Status Tab



### Report state of the block

Allows you to use the signal output port labeled **st**.

### Report memory value upon write event

Allows you to use the signal output port labeled **mem**.

### Report memory value upon read event

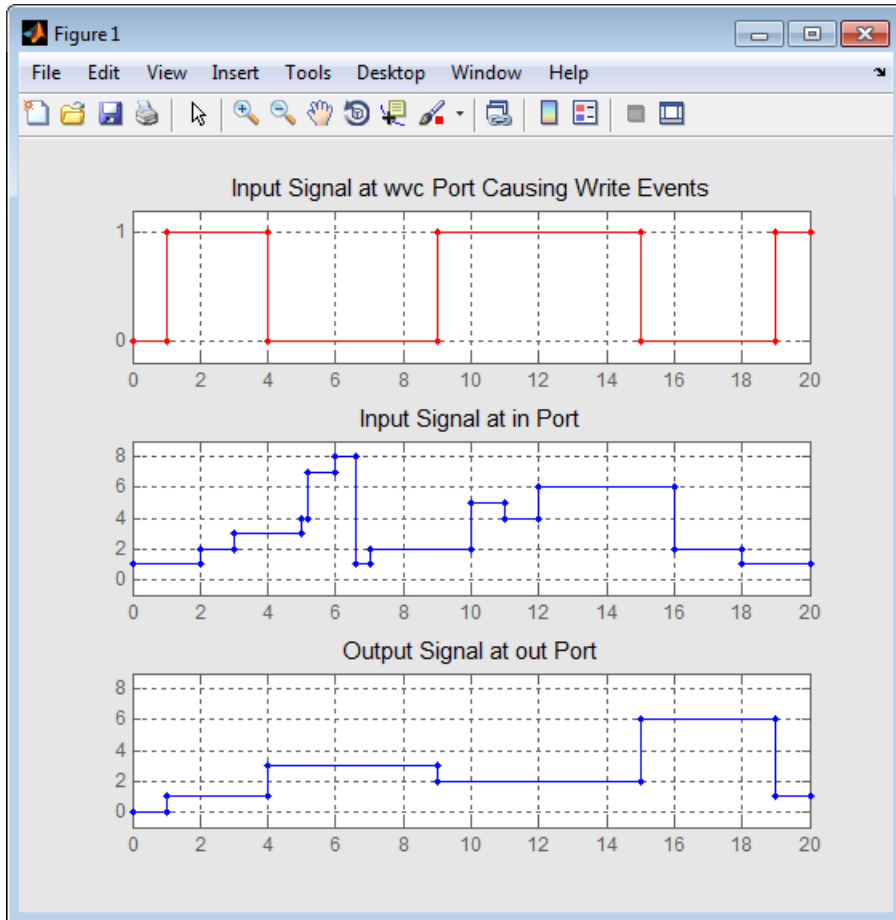
Allows you to use the signal output port labeled **out**.

## Examples

### Reading from Memory Upon Each Write Event

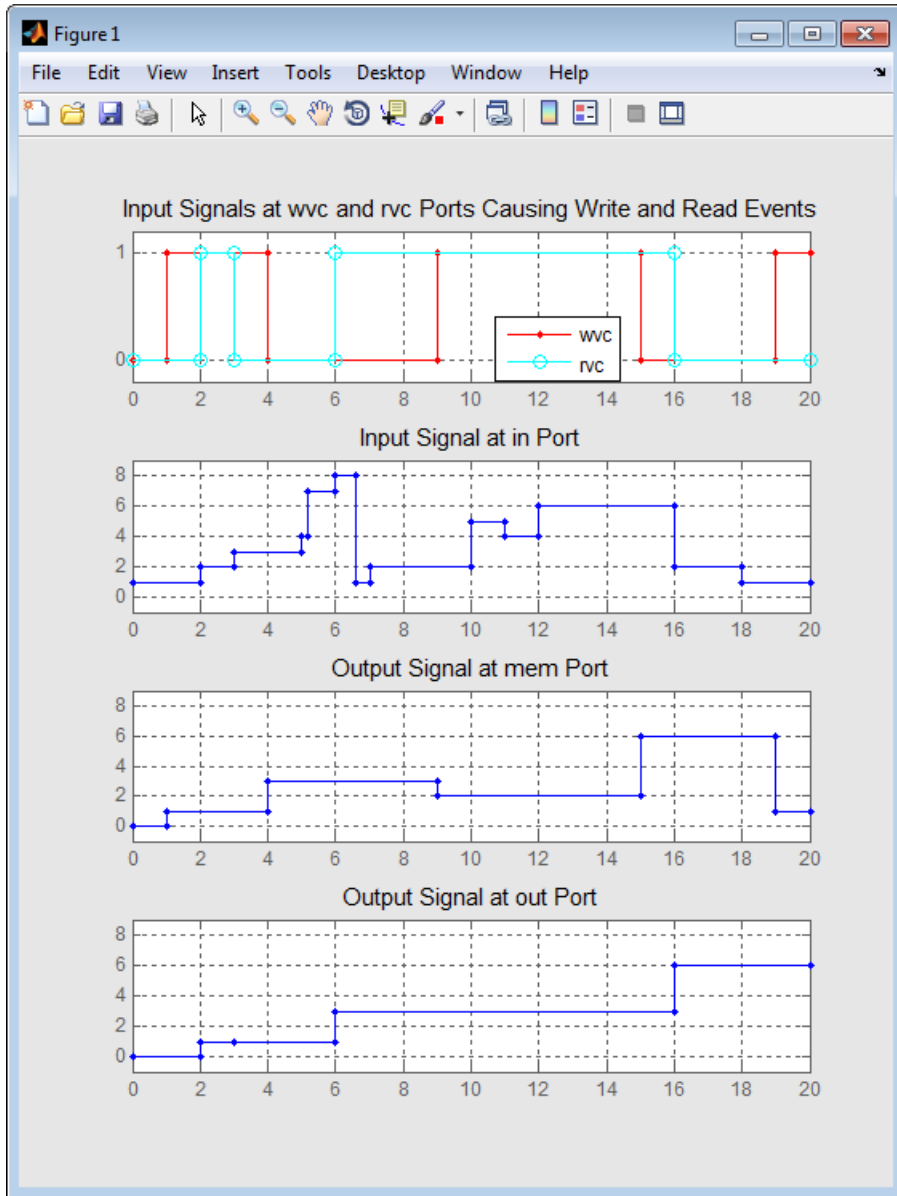
In the plot below, the output signal reflects values of the input signal upon each rising or falling value of the **wvc** signal. Between successive write events, the output signal

maintains the value from the most recent write event. Before the first write event, the output signal is 0 because of the initial memory value.



### Independent Read and Write Events

In the plot below, the **mem** signal reflects values of the input signal upon each rising or falling value of the **wvc** signal, while the **out** signal reflects values of the **mem** signal upon each rising or falling value of the **rvc** signal.



## See Also

Data Store Memory, Data Store Read, Data Store Write

**Introduced before R2006a**

## Signal Scope (Obsolete)

Plot data from signal

### Library

SimEvents Sinks

### Description



This block creates a plot using data from an event-based signal. The data for the vertical axis comes from the signal connected to the block's signal input port labeled **in**.

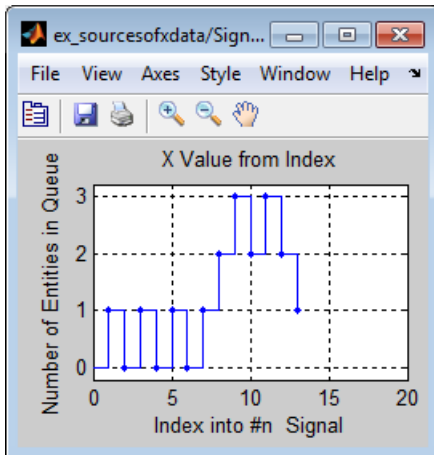
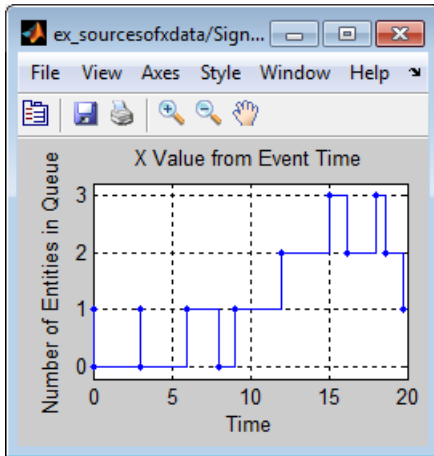
The **Plot type** parameter on the **Plotting** tab determines whether and how the block connects the points that it plots.

### Selecting Data for the Horizontal Axis

Use the **X value from** parameter to select the type of data for the horizontal axis. The table below describes the choices.

Source of X Data	Description of Plot
Event time	Plot of the <b>in</b> signal versus simulation time. For example, you might use this option to see how the length of a queue changes over time.
Index	Plot of the <b>in</b> signal's successive values against a horizontal axis that represents the index of the values. The signal's first value during the simulation has an index of 1, the signal's second value has an index of 2, and so on. For example, you might use this option for a signal that has zero-duration values, to help determine the exact sequence among values that the signal assumes simultaneously.

The figures below illustrate the different sources of data for the horizontal axis. The plots look similar, except that the second plot has uniform horizontal spacing rather than time-based spacing between successive points.



## Ports

### Signal Input Ports

Label	Description
in	Signal containing data for the Y axis. This signal must be an event-based signal.

### Signal Output Ports

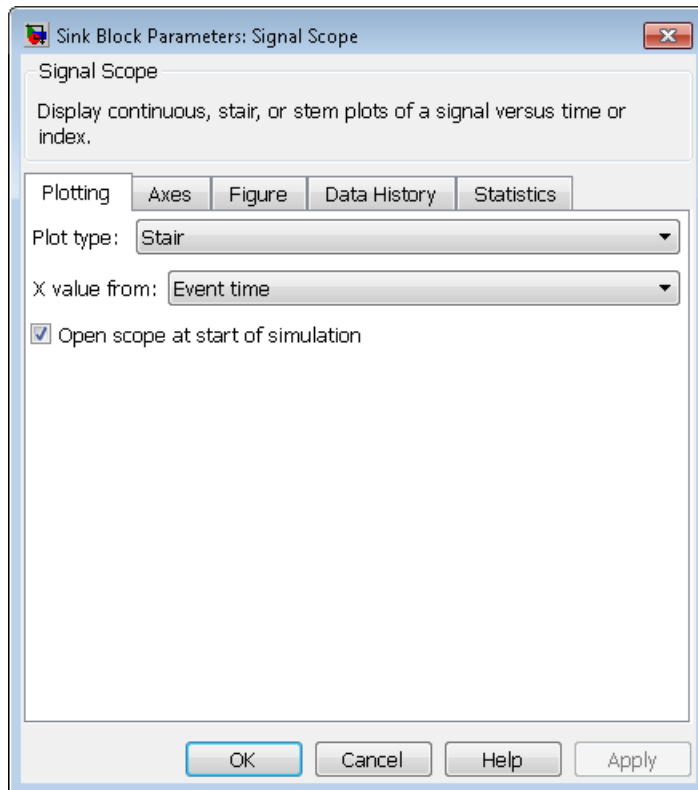
Label	Description
#c	Number of points the block has plotted.

The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

## Dialog Box

To open the block dialog box, click the Parameters toolbar button in the plot window.

## Plotting Tab





### Plot type

The presentation format for the data.

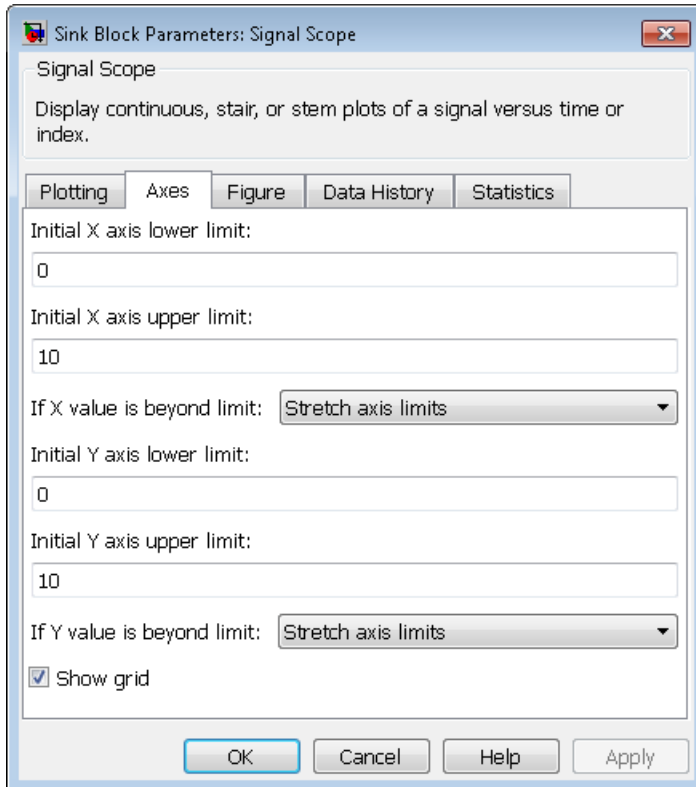
### X value from

Source of data for the plot's horizontal axis. See “Selecting Data for the Horizontal Axis” on page 2-284 for details.

### Open scope at start of simulation

Selecting this option causes the plot window to open when you start the simulation. If you clear this box, you can open the plot window by double-clicking the block icon.

## Axes Tab



### Initial X axis lower limit, Initial X axis upper limit

The interval shown on the X axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If X value is beyond limit** setting.

### **If X value is beyond limit**

Determines how the plot changes if one or more X values are not within the limits shown on the X axis.

### **Initial Y axis lower limit, Initial Y axis upper limit**

The interval shown on the Y axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If Y value is beyond limit** setting.

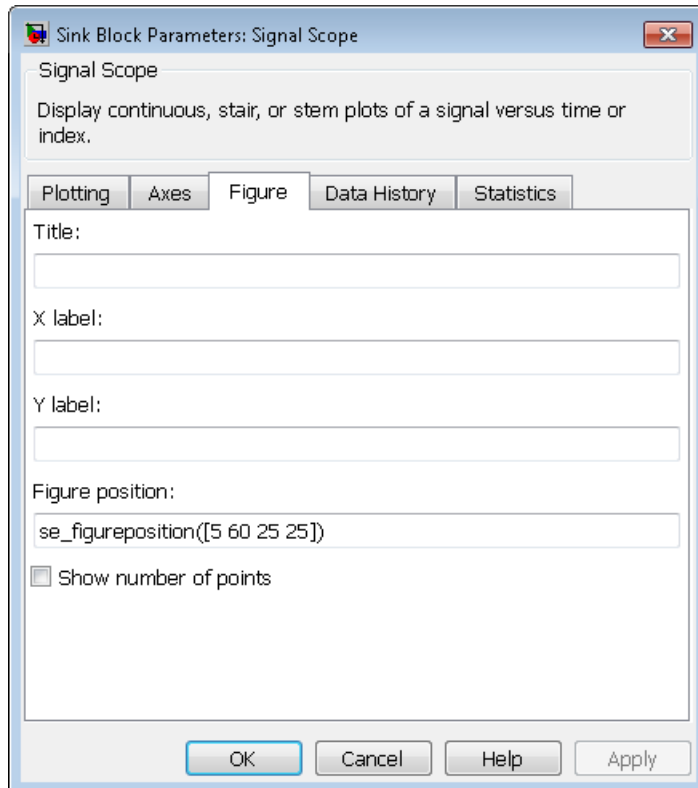
### **If Y value is beyond limit**

Determines how the plot changes if one or more values of the **in** signal are not within the limits shown on the Y axis.

### **Show grid**

Toggles the grid on and off.

## Figure Tab



### Title

Text that appears as the title of the plot, above the axes.

### Y label

Text that appears to the left of the vertical axis.

### X label

Text that appears below the horizontal axis.

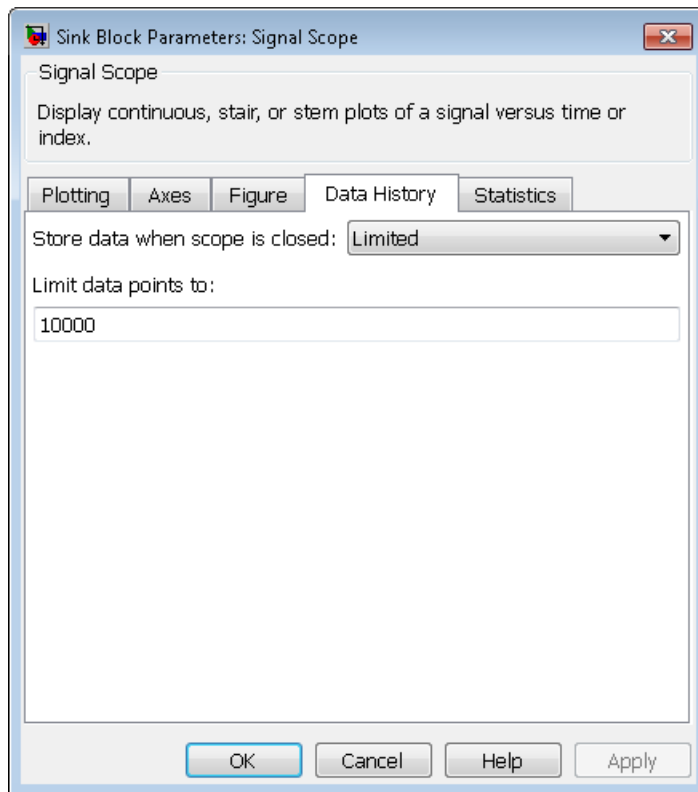
### Position

A four-element vector of the form `[left bottom width height]` specifying the position of the scope window. (0,0) is the lower left corner of the display.

### Show number of points

Displays the number of plotted points using an annotation in the plot window.

### Data History Tab



### Store data when scope is closed

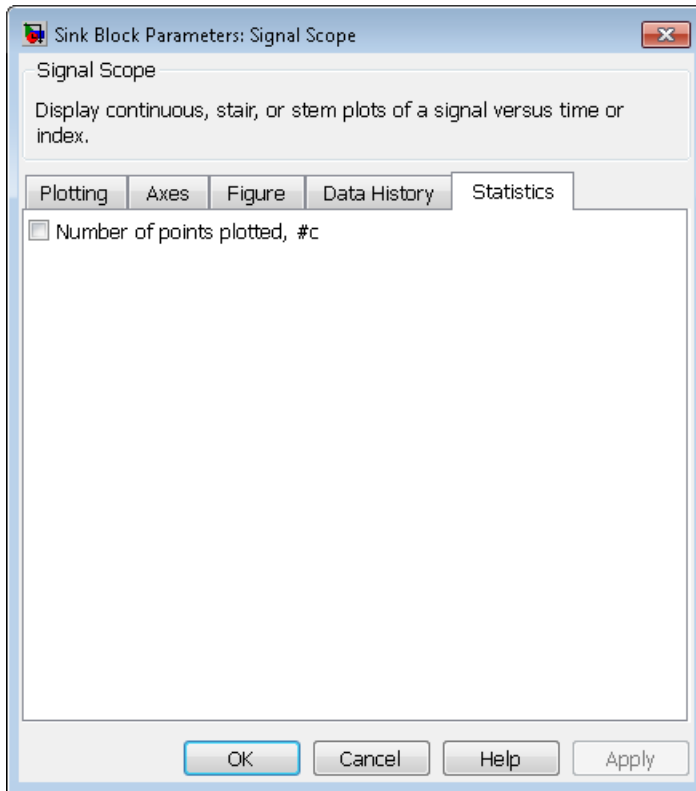
Select **Unlimited** to have the block cache all data for future viewing, **Limited** to cache a portion of the most recent data, and **Disabled** to avoid caching undisplayed data.

### Limit data points to

The number of data points the block caches, using the most recent data. You see this field only if you set **Store data when scope is closed** to **Limited**.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see the preceding table, “Signal Output Ports”.



### Number of points plotted

Allows you to use the signal output port labeled **#c**.

## Examples

- “Build a Discrete-Event Model”

## **See Also**

X-Y Signal Scope (Obsolete), Attribute Scope (Obsolete)

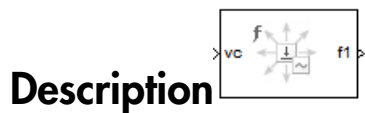
**Introduced before R2006a**

# Signal-Based Function-Call Generator (Obsolete)

Convert signal-based events into function calls

## Library

Generators/Function-Call Generators



This block converts a signal-based event or a function-call input into one or two function calls that you can use to invoke function-call subsystems, Stateflow blocks, or other blocks that accept function-call inputs. You specify the type of event the block translates and whether the block suppresses its output under certain conditions. You can also delay the output function calls by an amount of time that you specify via a parameter or an input signal.

## Criteria for Generating Function Calls

The primary criterion, based on the **Generate function call only upon** parameter, is a signal-based event or a function call. By default, the block generates a function call upon each event of the type you specify.

To generate up to two function calls upon each event, select **Generate optional f2 function call**. If the block generates function calls at both the **f1** and **f2** output ports, then it generates the **f1** call first and generates the **f2** call as a subsequent part of the same operation.

To make the **f1** or **f2** output function call contingent upon a secondary criterion, select **Suppress function call f1 if enable signal e1 is not positive** or **Suppress function call f2 if enable signal e2 is not positive**. The block acquires an additional signal input port, labeled **e1** or **e2**, to which you connect a control signal. If the control signal is zero or negative when the block is about to generate the function call, then the block suppresses the function call. The **e1** and **e2** ports operate independently of each other as secondary criteria for their respective function-call output ports.

## Ports

### Signal Input Ports

Label	Description
t	The delay, in seconds, between the input event and the output function call. A positive value schedules the function call in the future, while a value of zero schedules the function call at the current simulation time. This signal must be an event-based signal. You see this port only if you select <b>Resolve simultaneous signal updates according to event priority</b> , and then set <b>Function-call delay from</b> to Signal port t.
ts	When this signal has an update, the primary criterion is satisfied. This signal must be an event-based signal. You see this port only if you set <b>Generate function call only upon</b> to Sample time hit from port ts.
tr	When this signal has a rising or falling edge, depending on the <b>Trigger type</b> parameter, the primary criterion is satisfied. This signal must be an event-based signal. You see this port only if you set <b>Generate function call only upon</b> to Trigger from port tr.
vc	When this signal increases or decreases, depending on the <b>Type of change in signal value</b> parameter, the primary criterion is satisfied. This signal must be an event-based signal. You see this port only if you set <b>Generate function call only upon</b> to Change in signal from port vc.
fcn	When this signal carries a function call, the primary criterion is satisfied. This signal must be an event-based function call. You see this port only if you set <b>Generate function call only upon</b> to Function call from port fcn. Do not connect this port to an output port from the same instance of this block.
e1	When this signal is 0 or negative, the block does not generate a function call at the <b>f1</b> output port. This signal must be an event-based signal. You see this input port only if you select <b>Suppress function call f1 if enable signal e1 is not positive</b> .
e2	When this signal is 0 or negative, the block does not generate a function call at the <b>f2</b> output port. This signal must be an event-based signal. You see this input port only if you select <b>Suppress function call f2 if enable signal e2 is not positive</b> .

### Signal Output Ports



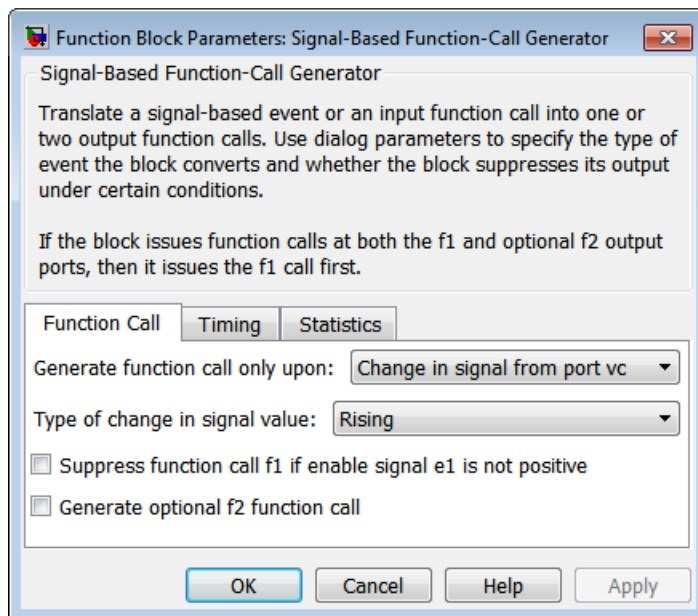
Label	Description	Order of Update
<b>f1</b>	Function call, possibly contingent on <b>e1</b> input signal	1
<b>f2</b>	Function call, possibly contingent on <b>e2</b> input signal	2
<b>#f1</b>	Number of function calls the block has generated at the <b>f1</b> port during the simulation	3
<b>#f2</b>	Number of function calls the block has generated at the <b>f2</b> port during the simulation	3

Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

The initial output value for the numerical signals, which is in effect from the start of the simulation until the first update by the block, is 0.

## Dialog Box

### Function Call Tab



#### Generate function call only upon

The primary criterion for determining when the block generates a function call. Optional secondary criteria are established by the **Suppress function call...** parameters below.

#### Trigger type, Type of change in signal value

**Trigger type** determines whether rising, falling, or either type of trigger edge causes the block to generate a function call. You see this field only if you set **Generate function call only upon** to **Trigger** from port **tr**.

**Type of change in signal value** determines whether rising, falling, or either type of value change causes the block to generate a function call. You see this field only if you set **Generate function call only upon** to **Change in signal** from port **vc**.

#### Suppress function call f1 if enable signal e1 is not positive

Selecting this option causes **f1** function calls to be contingent upon a positive value at the **e1** signal input port.

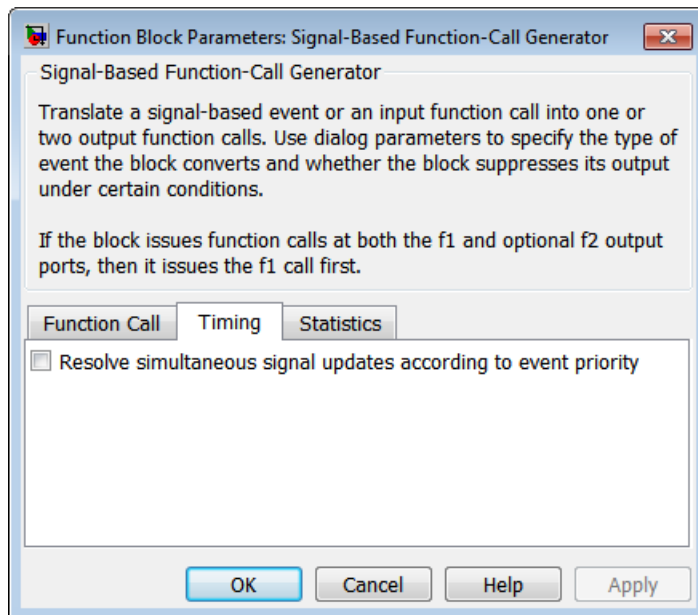
### Generate optional **f2** function call

Selecting this option causes the block to generate a function call at the optional **f2** output port when appropriate criteria are satisfied.

### Suppress function call **f2** if enable signal **e2** is not positive

Selecting this option causes **f2** function calls to be contingent upon a positive value at the **e2** signal input port. You see this field only if you select **Generate optional **f2** function call**.

## Timing Tab



### Resolve simultaneous signal updates according to event priority

Select this option to control the sequencing of the function-call event, relative to other simultaneous events in the simulation. If you do not select this option, the application issues the function call immediately upon detecting the signal-based event that causes it.

---

**Note:** If this block has both a function-call input and a signal input, you might need to select this option to prevent latency in the signal.

---

### Event priority

The priority of the function-call event, relative to other simultaneous events in the simulation.

Use of this parameter depends on the following:

- You see this field only if you select **Resolve simultaneous signal updates according to event priority**.
- If you select the configuration parameter **Prevent duplicate events on multiport blocks and branched signals** in your model, the software uses the **Event priority** parameter to sort blocks in the model. In this case, the software does not schedule an event that you can view on the event calendar.

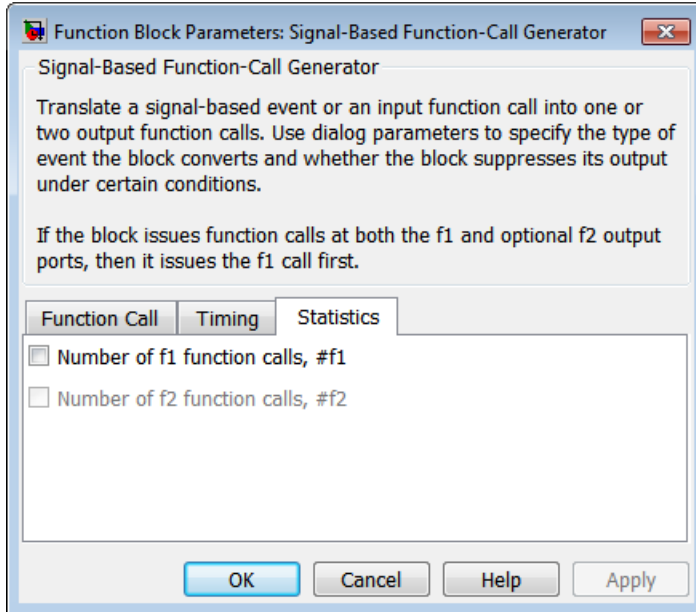
### Function-call delay from

Determines whether the delay between the input event and the output function call is computed from a parameter in this dialog box or from an input signal. You see this field only if you select **Resolve simultaneous signal updates according to event priority**.

### Function-call time delay

The delay, in seconds, between the input event and the output function call. A positive value schedules the function call in the future, while a value of zero schedules the function call at the current simulation time. You see this field only if you select **Resolve simultaneous signal updates according to event priority**, and then set **Function-call delay from** to Dialog.

## Statistics Tab



### Number of f1 function calls

Allows you to use the signal output port labeled **#f1**.

### Number of f2 function calls

Allows you to use the signal output port labeled **#f2**. This field is active only if you select **Generate optional f2 function call** on the **Function Call** tab of this dialog box.

## See Also

Entity Departure Function-Call Generator (Obsolete)

Introduced in R2011b

## Signal-Based Function-Call Event Generator (Obsolete)

Generate function-call events in response to signal-based events

### Library

Generators / Event Generators




---

**Note:** The Signal-Based Function-Call Event Generator block will be removed in a future release. Use the Signal-Based Function-Call Generator (Obsolete) block instead.

---

This block generates an output function call corresponding to each signal-based event or input function call. You specify the type of event the block responds to. You can use the function call to invoke function-call subsystems, Stateflow blocks, or other blocks that accept function-call inputs.

This block is similar to the Signal-Based Function-Call Generator (Obsolete) block, which offers more flexibility.

### Ports

#### Signal Input Ports

Label	Description
ts	When this signal has an update, the block generates a function call. This signal must be an event-based signal. You see this port only if you set <b>Generate function call only upon</b> to Sample time hit from port ts.

Label	Description
<b>tr</b>	When this signal satisfies the specified trigger criteria, the block generates a function call. This signal must be an event-based signal. You see this port only if you set <b>Generate function call only upon</b> to <b>Trigger</b> from port <b>tr</b> .
<b>vc</b>	When this signal satisfies the specified value-change criteria, the block generates a function call. This signal must be an event-based signal. You see this port only if you set <b>Generate function call only upon</b> to <b>Change in signal</b> from port <b>vc</b> .
<b>fcn</b>	When this signal carries a function call, the block generates a function call. This signal must be an event-based function call. You see this port only if you set <b>Generate function call only upon</b> to <b>Function call</b> from port <b>fcn</b> . Do not connect this port to an output port from the same instance of this block.

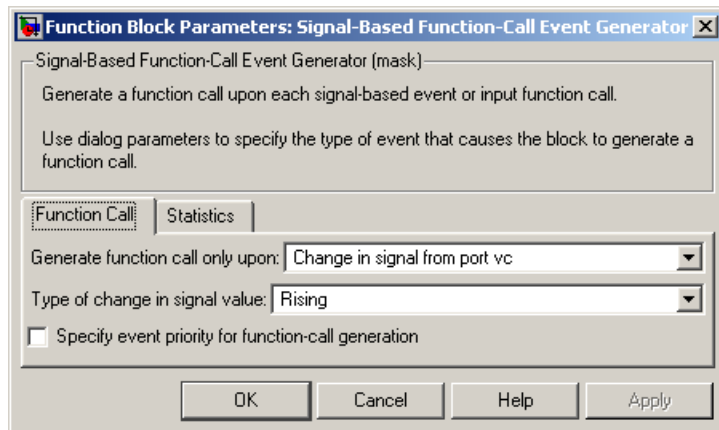
### Signal Output Ports

Label	Description	Order of Update
<b>f1</b>	Function-call signal.	1
<b>#f1</b>	Number of function calls the block has generated during the simulation.	2

The initial output value for the numerical signals, which is in effect from the start of the simulation until the first update by the block, is 0.

## Dialog Box

### Function Call Tab



#### Generate function call only upon

The primary criterion for determining when the block generates a function call. Optional secondary criteria are established by the **Suppress function call...** parameters below.

#### Trigger type

Determines whether rising, falling, or either type of trigger edge causes the block to generate a function call. You see this field only if you set **Generate function call only upon** to **Trigger from port tr**.

#### Type of change in signal value

Determines whether rising, falling, or either type of value change causes the block to generate a function call. You see this field only if you set **Generate function call only upon** to **Change in signal from port vc**.

#### Resolve simultaneous signal updates according to event priority

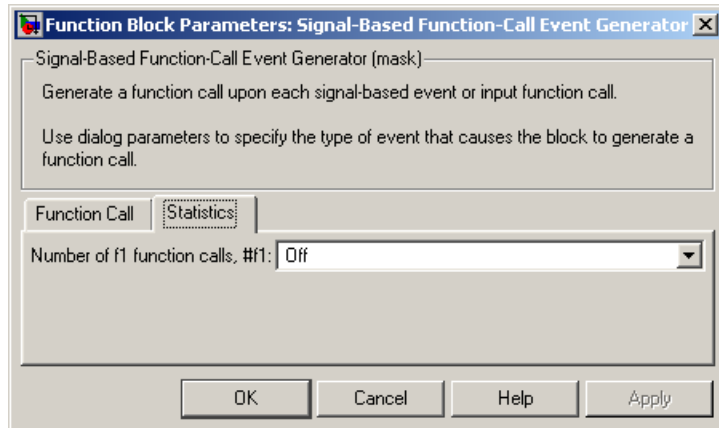
Select this option to control the sequencing of the function-call event, relative to other simultaneous events in the simulation. If you do not select this option, the application issues the function call immediately upon detecting the signal-based event that causes it.

#### Event priority



The priority of the function-call event, relative to other simultaneous events in the simulation. You see this field only if you select **Resolve simultaneous signal updates according to event priority**.

## Statistics Tab



### Number of f1 function calls

Allows you to use the signal output port labeled **#f1**.

## See Also

Signal-Based Function-Call Generator (Obsolete)

Introduced before R2006a

## Single Server (Obsolete)

Serve one entity for period of time

### Library

Servers



This block serves one entity for a period of time, and then attempts to output the entity through the **OUT** port. If the **OUT** port is blocked, then the entity stays in this block until the port becomes unblocked. If an entity in this block is scheduled to time out, then it might depart prematurely via the optional **TO** port.

You specify the service time, which is the duration of service, via a parameter, attribute, or signal, depending on the **Service time from** parameter value. The block determines the service time for an entity upon its arrival. Service times are assumed to be specified in seconds.

---

**Note:** If you specify the service time via an event-based signal, be sure that its updates occur before the entity arrives.

---

The block permits preemption if you select **Permit preemption based on attribute**. In this case, an entity in the server can depart early via the **P** port. Preemption occurs only if attributes of the current entity and the entity attempting to arrive satisfy specified criteria.

When the block does not permit preemption, the **IN** port is unavailable whenever this block stores an entity. In this case, the **IN** port becomes available when the entity departs.

## Ports

### Entity Input Ports

Label	Description
IN	Port for arriving entities, which will be served.

### Signal Input Ports

Label	Description
t	Service time, in seconds, for a newly arrived entity. This signal must be an event-based signal. You see this port only if you set <b>Service time from</b> to <b>Signal port t</b> .

### Entity Output Ports

Label	Description
OUT	Port for departing entities that have completed their service time, have not timed out while in this block, and have not been preempted.
P	Port for entities that have been preempted by an arriving entity. This port must not be blocked at the time of preemption.
TO	Port for entities that time out while in this block. You see this port only if you select <b>Enable TO port for timed-out entities</b> . This port must not be blocked when an entity attempts to depart here.

### Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block via the <b>OUT</b> port since the start of the simulation.	After entity departure via the <b>OUT</b> port	4
#n	Number of entities currently in the block, either 0 or 1.	After entity arrival and after entity departure via the <b>OUT</b> or <b>TO</b> port	3

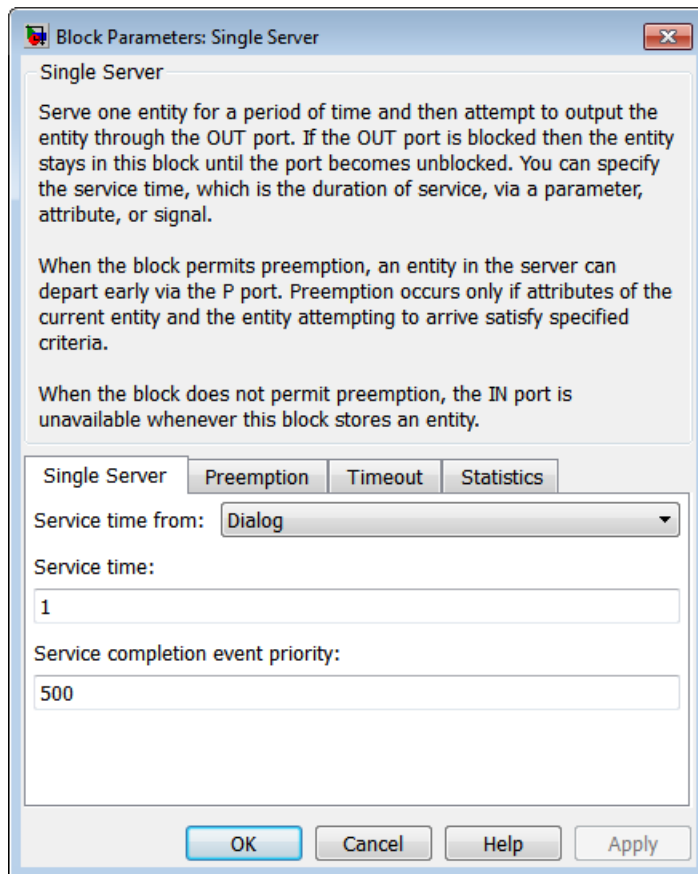
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#p	Number of entities that have been preempted from this block since the start of the simulation.	After entity departure via the <b>P</b> port	4
pe	<p>A value of 1 indicates that the block stores at least one entity that has tried and failed to depart. In that case, the entity is a pending entity.</p> <p>A value of 0 indicates that the block does not store any pending entities.</p>	<p>Sample time hit of 1 occurs after the block stores an entity that has tried and failed to depart.</p> <p>Sample time hit of 0 occurs after the departure of the pending entity via any port.</p>	1
w	Sample mean of the waiting times in this block for all entities that have departed from the <b>OUT</b> or <b>TO</b> port. An entity's waiting time might exceed its service time if the <b>OUT</b> port is blocked when the entity completes service.	After entity departure via the <b>OUT</b> or <b>TO</b> port	2
util	Utilization of the server, which is the fraction of simulation time spent storing an entity. At $T=0$ , the utilization is 0 or 1 depending on whether the server contains an entity.	Performance considerations cause the block to suppress signal updates until specific occurrences cause updates. In <b>On</b> mode, updates occur after an entity departure via the <b>OUT</b> or <b>TO</b> port, and after an entity arrival. In <b>Upon stop or pause</b> mode, updates occur when the simulation stops or pauses.	2
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the <b>TO</b> port	4

Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

### Single Server Tab



**Service time from**

Determines whether the service time is computed from a parameter in this dialog box, an input signal, or an attribute of the entity being served.

**Service time**

The service time, in seconds, for all entities. You see this field only if you set **Service time from** to **Dialog**.

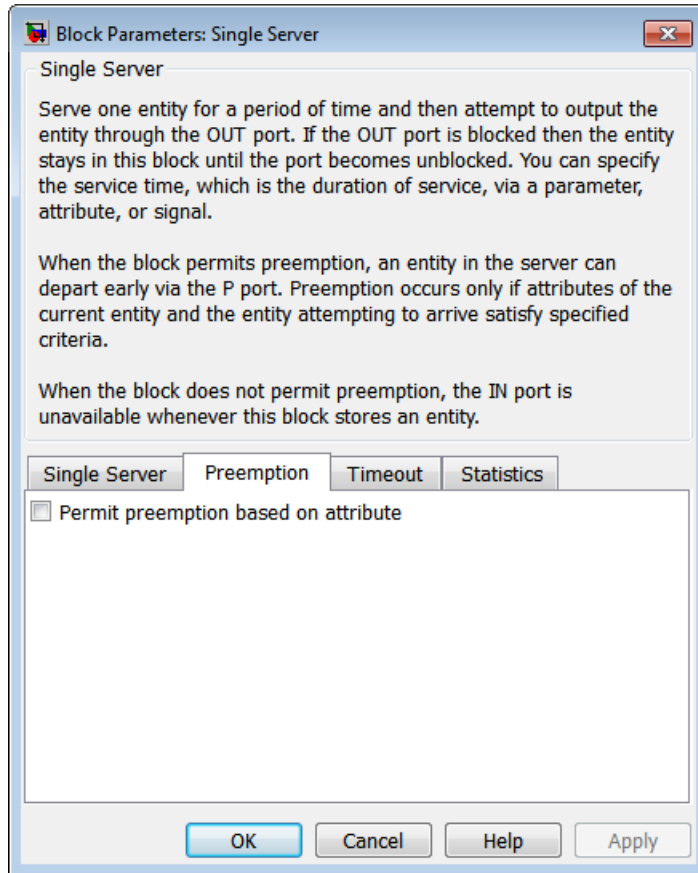
**Attribute name**

The name of the attribute whose value the block uses as the service time for an entity. You see this field only if you set **Service time from** to **Attribute**.

**Service completion event priority**

The priority of the service completion event, relative to other simultaneous events in the simulation.

## Preemption Tab



### Permit preemption based on attribute

If you select this option, the block can replace an entity by a higher priority entity. Otherwise, the block never permits new arrivals when it is storing an entity. Selecting this option also clears the **Average wait, w** check box on the **Statistics** tab and makes that parameter unavailable.

### Sorting attribute name

The block uses this attribute to determine whether a new entity can preempt the one in the server. You see this field only if you select **Permit preemption based on attribute**.

### **Sorting direction**

Preemption occurs when the arriving entity has a strictly smaller (**Ascending**) or strictly larger (**Descending**) value of the attribute named above, compared to the attribute value of the entity in the server. You see this field only if you select **Permit preemption based on attribute**.

### **Write residual service time to attribute**

If you select this option, a preemption event causes the block to set an attribute in the preempted entity. The attribute value is the remaining service time the entity would have required if it had not been preempted. You see this field only if you select **Permit preemption based on attribute**.

### **Residual service time attribute name**

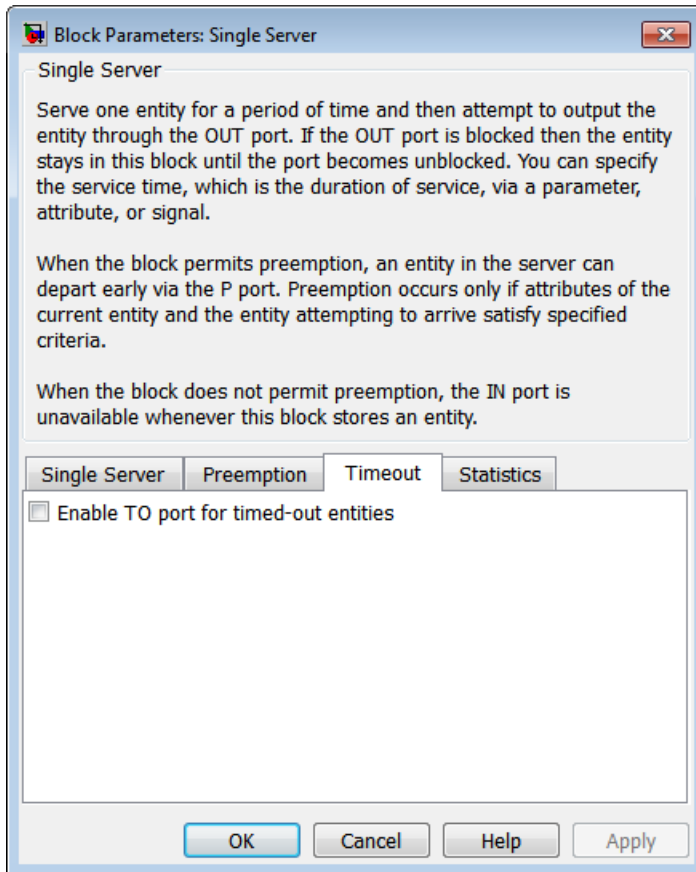
The name of the attribute the block uses when recording the residual service time of entities. You see this field only if you select **Write residual service time to attribute**.

### **Create attribute if not present**

Selecting this option enables the block to define a new attribute for the residual service time. Otherwise, the block issues an error if the attribute named above does not already exist. You see this field only if you select **Write residual service time to attribute**.



## Timeout Tab

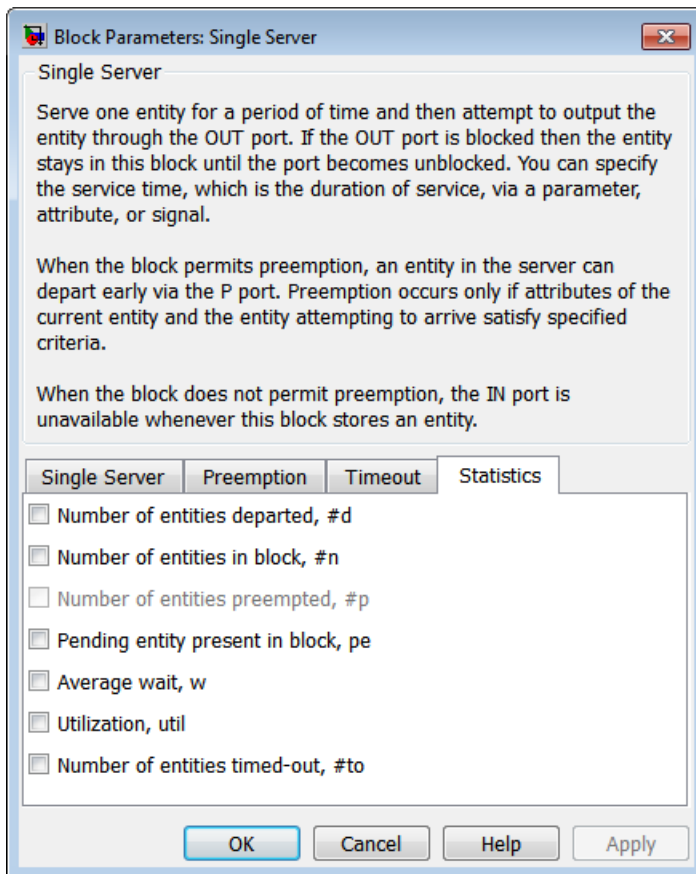


### Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the Schedule Timeout (Obsolete) block.

## Statistics Tab

These parameters determine whether certain ports produce data throughout the simulation, produce data only when you stop or pause the simulation, or are omitted from the block. For descriptions of the affected ports, see the preceding table, “Signal Output Ports”.



### Number of entities departed

Allows you to use the signal output port labeled **#d**.

### Number of entities in block

Allows you to use the signal output port labeled **#n**.

### Number of entities preempted

Allows you to use the signal output port labeled **#p**. This field is available only if you select the **Permit preemption based on attribute** option on the **Preemption** tab.

### Pending entity present in block

Allows you to use the signal output port labeled **pe**.

### Average wait

Allows you to use the signal output port labeled **w**. This field is available only if you clear the **Permit preemption based on attribute** option on the **Preemption** tab.

### Utilization

Allows you to use the signal output port labeled **util**.

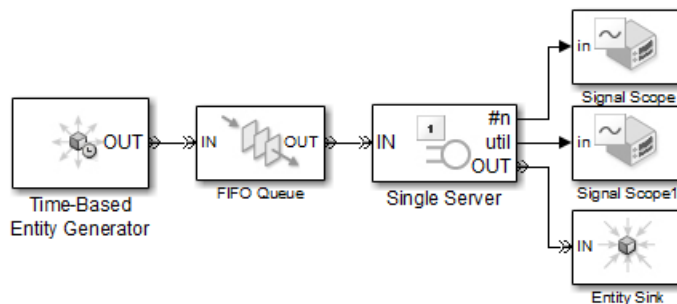
### Number of entities timed out

Allows you to use the signal output port labeled **#to**.

## Examples

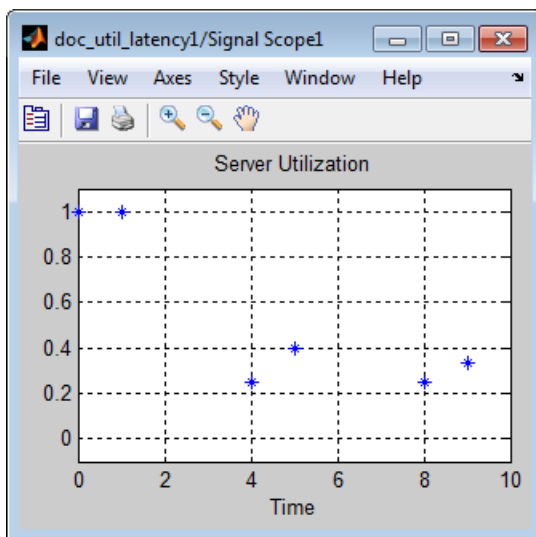
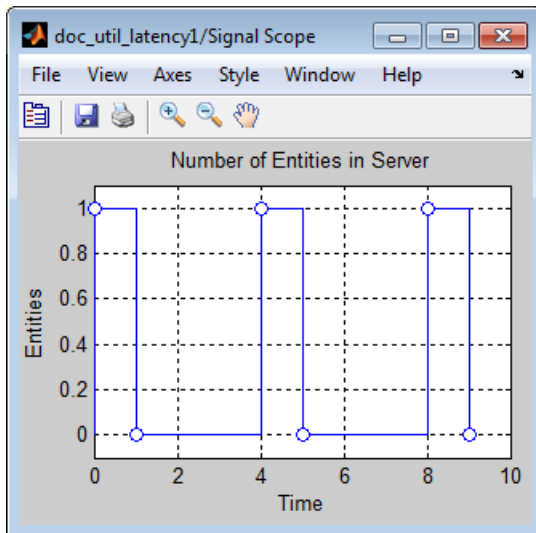
- “Build a Discrete-Event Model”
- “Constructs Involving Queues and Servers”

The following example illustrates the timing of updates of the **util** signal, as described in Signal Output Ports.

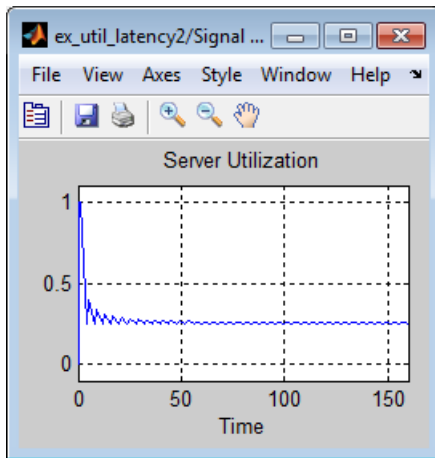


The server has idle periods that reduce its utilization. However, the server block recomputes the **util** signal only when the number of entities in the server changes. While

the definition of utilization says that the utilization is less than 1 at time 3, the **util** signal remains at its previous value of 1 until the next entity arrives at time 4.



In a longer simulation, the differences in the value of **util** compared to its theoretical definition become less pronounced.



## See Also

N-Server (Obsolete), Infinite Server (Obsolete)

“Write Events Actions”

**Introduced before R2006a**

## Start Timer (Obsolete)

Associate named timer to each arriving entity independently and start timing

### Library

Timing



This block associates a named timer to each arriving entity independently and starts the timer. If the entity was previously associated with a timer of the same name, then the block either continues or restarts that timer, depending on your setting for the **If timer has already started** parameter; the **Warn and continue** option can be helpful for debugging or preventing modeling errors. Other timers, if any, associated with the arriving entity are unaffected.

This block works with the **Read Timer (Obsolete)** block. To read the value of the timer named in this block, reference the timer name in the **Read Timer** block.

### Ports

#### Entity Input Ports

Label	Description
IN	Port for arriving entities.

#### Entity Output Ports

Label	Description
OUT	Port for departing entities, which have named timers attached to them.

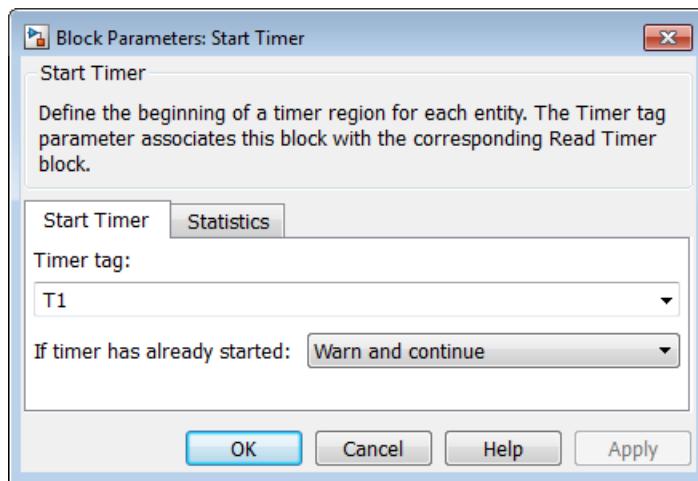
#### Signal Output Ports

Label	Description	Time of Update When Statistic Is On
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure

The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

## Dialog Box

### Start Timer Tab



#### Timer tag

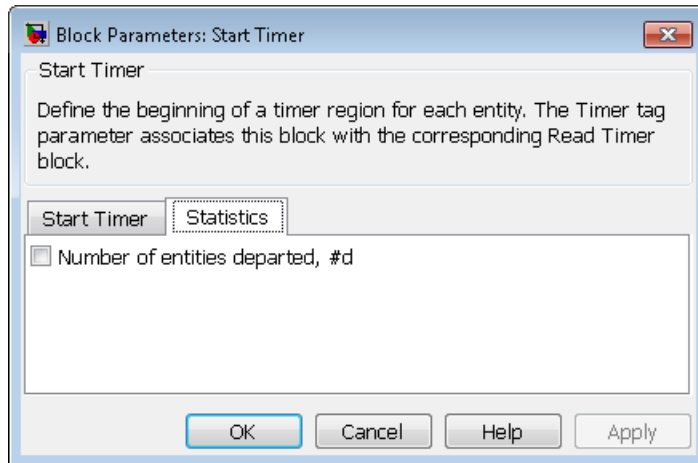
Name of the timer to associate with each entity. Enter a new timer tag, or restart a previous timer by choosing it in the drop-down list.

#### If timer has already started

Behavior of the block if an arriving entity already has a timer with the specified timer tag.

### Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports.



#### Number of entities departed

Allows you to use the signal output port labeled **#d**.

### See Also

Read Timer (Obsolete)

Introduced before R2006a



# Time-Based Entity Generator (Obsolete)

Generate entities using intergeneration times from signal or statistical distribution

## Library

Generators / Entity Generators



This block is designed to generate entities using intergeneration times that satisfy criteria that you specify. The intergeneration time is the time interval between two successive generation events.

Intergeneration Times	Value of Generate entities upon Parameter
Distributed according to various parameters in the block dialog box	Intergeneration time from dialog
Specified using an input signal that the block reads at the start of the simulation and each time it generates an entity	Intergeneration time from port t

## Responding to Blockage at the Entity Output Port

You can choose how this block responds when it generates an entity that the subsequent entity input port is not available to accept:

- If you set **Response when blocked** to **Error**, the simulation halts with an error message.
- If you set **Response when blocked** to **Pause generation**, this block holds the entity, which becomes a pending entity. The block does not schedule another entity generation event yet. The **Response when unblocked** parameter determines what the block does next:

- If you set **Response when unblocked** to **Immediate restart**, after this block learns that the subsequent port is available, the pending entity departs. After the pending entity departs, this block schedules the generation of the next entity.
- If you set **Response when unblocked** to **Delayed restart**, upon learning that the subsequent port is available, this block schedules an event of type **DelayedRestart**. The event time is the current time plus the same intergeneration time the block used when generating the pending entity. When the block executes the event, the pending entity attempts to depart.

Use the **Delayed restart** option if you want to:

- Keep the arrival process memoryless, when **Distribution** is **Exponential**.
- Prevent correlation among multiple instances of this block if they become unblocked simultaneously.

For an example, see “Example: Responding to Blockage” on page 2-326.

## Ports

### Signal Input Ports

Label	Description
t	Time interval between generation events of the current entity and the next entity. The block reads the value after the current entity departs and the block updates its output signals, if any. If you do not select <b>Generate entity at simulation start</b> , then the block also reads the value of this signal at the start of the simulation. This signal must be an event-based signal. You see this port only if you set <b>Generate entities upon</b> to <b>Intergeneration time</b> from port t.

### Entity Output Ports

Label	Description
OUT	Port through which generated entities depart.

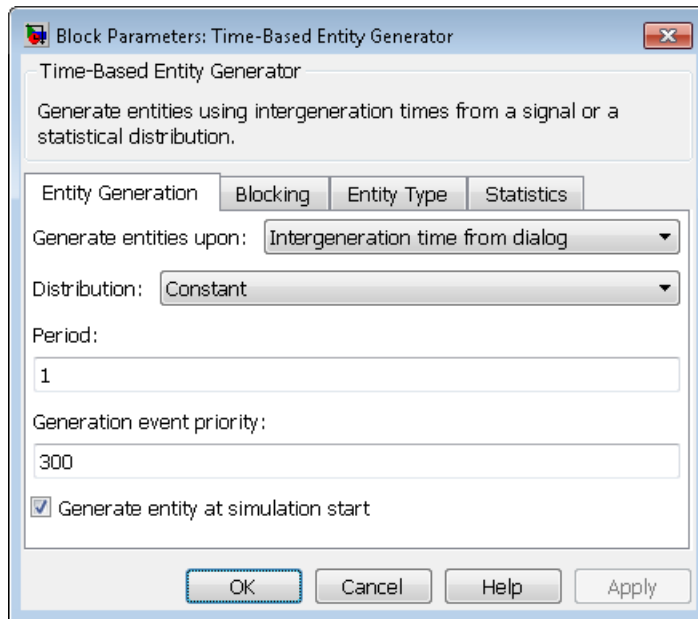
### Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	3
pe	<p>A value of 1 indicates that the block stores an entity that has tried and failed to depart. In that case, the entity is a pending entity.</p> <p>A value of 0 indicates that the block does not store any pending entities.</p>	<p>Sample time hit of 1 occurs after the block stores an entity that has tried and failed to depart.</p> <p>Sample time hit of 0 occurs after the departure of the pending entity.</p>	1
w	Average interdeparture time, in seconds, for all pairs of successive entities that have departed from this block. The signal value is 0 before the second entity departure.	After entity departure	2

The initial output value, which is in effect from the start of the simulation until the first update by the block, is 0 for all signals.

## Dialog Box

### Entity Generation Tab



#### Generate entities upon

Determines where the block gets instructions about when to generate entities.

#### Distribution

The statistical distribution of intergeneration times. You see this field only if you set **Generate entities upon** to Intergeneration time from dialog.

#### Period

The time interval between entity generations, in seconds. You see this field only if you set **Generate entities upon** to Intergeneration time from dialog and set **Distribution** to Constant.

#### Initial seed

A nonnegative integer that initializes the random number generator. You see this field only if you set **Generate entities upon** to Intergeneration time from dialog and set **Distribution** to Uniform or Exponential.

**Minimum**

The lower endpoint, in seconds, of the interval over which the distribution is uniform. This field appears only if you set **Generate entities upon** to Intergeneration time from dialog and set **Distribution** to Uniform.

**Maximum**

The upper endpoint, in seconds, of the interval over which the distribution is uniform. This field appears only if you set **Generate entities upon** to Intergeneration time from dialog and set **Distribution** to Uniform.

**Mean**

The expected value of the exponential distribution. You see this field only if you set **Generate entities upon** to Intergeneration time from dialog and set **Distribution** to Exponential.

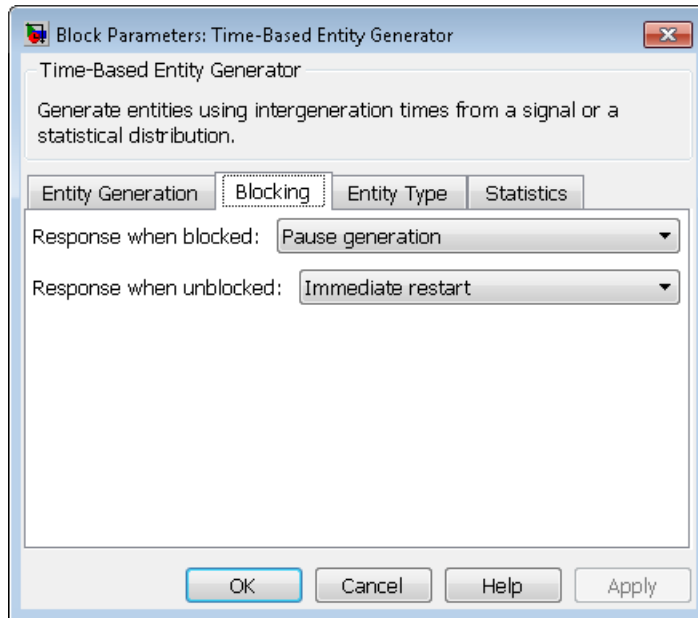
**Generation event priority**

The priority of the entity-generation event, relative to other simultaneous events in the simulation.

**Generate entity at simulation start**

If you select this option, the block generates the first entity when the simulation begins and the second entity at the first intergeneration time. Otherwise, the block generates the first entity at the first intergeneration time.

## Blocking Tab



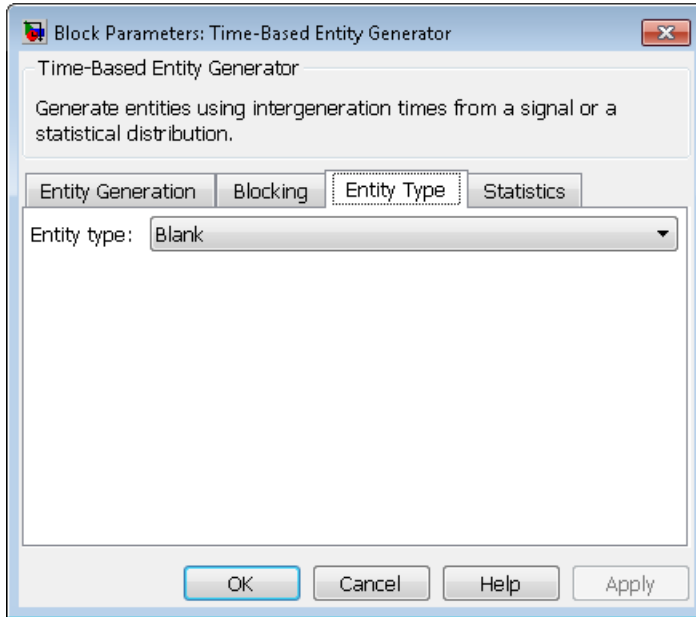
### Response when blocked

Determines how the block responds if a generated entity cannot depart immediately because the entity input port of the subsequent block is unavailable; see “Responding to Blockage at the Entity Output Port” on page 2-319.

### Response when unblocked

Determines entity generation behavior if the entity input port of the subsequent block is available again after a prior blockage; see “Responding to Blockage at the Entity Output Port” on page 2-319.

## Entity Type Tab

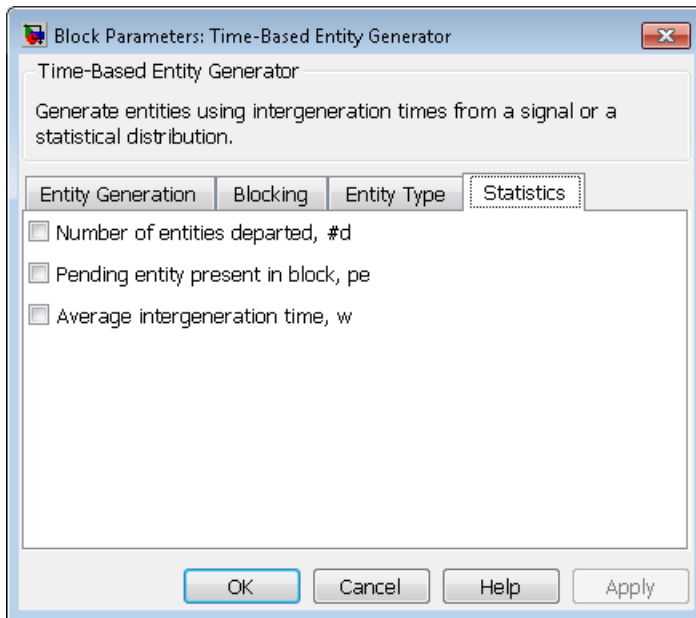


### Entity type

The blank type includes no attributes. The standard type includes attributes called Priority and Count, with default values of 10 and 0, respectively.

### Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports.



### **Number of entities departed**

Allows you to use the signal output port labeled **#d**.

### **Pending entity present in block**

Allows you to use the signal output port labeled **pe**.

### **Average intergeneration time**

Allows you to use the signal output port labeled **w**.

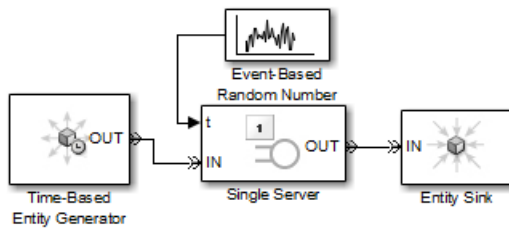
## **Examples**

- “Build a Discrete-Event Model”

### **Example: Responding to Blockage**

To illustrate the blockage options, consider a **Time-Based Entity Generator** block followed by a **Single Server** block, then followed by an **Entity Sink** block.





Suppose the block configurations have these characteristics:

- The entity generator has **Response when blocked** set to **Pause** generation.
- The entity generator generates the first entity at  $T=1$  and uses an intergeneration time of 1 s.
- The service times for the first three entities in the server are 1.5, 2.2, and 1.8.

The following tables indicate how the **Response when unblocked** values affect the behavior in the simulation.

#### Immediate Restart

Time (s)	Behavior
1	Entity generator generates and outputs the first entity to the server. The entity input port of the server becomes unavailable. The first entity is in service until $T=1+1.5=2.5$ .
2	Entity generator generates the second entity and holds it because the <b>OUT</b> port is blocked.
2.5	First entity departs from the server. The entity input port of the server becomes available and the second entity advances from the entity generator to the server. The entity input port of the server then becomes unavailable. The second entity is in service until $T=2.5+2.2=4.7$ . The entity generator schedules the next generation for $T=2.5+1=3.5$ .
3.5	Entity generator generates the third entity, and holds it because the <b>OUT</b> port is blocked.
4.7	Second entity departs from the server. The entity input port of the server becomes available and the third entity advances from the entity generator to the server. The entity input port of the server then becomes unavailable. The third entity is in service until $T=4.7+1.8=6.5$ . The entity generator schedules the next generation for $T=4.7+1=5.7$ .

### Delayed Restart

Time (s)	Behavior
1	Entity generator generates the first entity. The entity advances to the server. The entity input port of the server becomes unavailable. The entity is in service until $T=1+1.5=2.5$ .
2	Entity generator generates the second entity. The entity becomes a pending entity because the <b>OUT</b> port is blocked.
2.5	First entity departs from the server. The entity input port of the server becomes available. The entity generator schedules a delayed restart event for the second entity at $T=2.5+1=3.5$ .
3.5	The second entity advances from the entity generator to the server. The entity input port of the server then becomes unavailable. The second entity is in service until $T=3.5+2.2=5.7$ . The entity generator schedules the next generation for $T=3.5+1=4.5$ .
4.5	Entity generator generates the third entity. The entity becomes a pending entity because the <b>OUT</b> port is blocked.
5.7	Second entity departs from the server. The entity input port of the server becomes available. The entity generator schedules a delayed restart event for the third entity at $T=5.7+1=6.7$ .
6.7	The third entity advances from the entity generator to the server. The entity input port of the server then becomes unavailable. The third entity is in service until $T=6.7+1.8=8.5$ . The entity generator schedules the next generation for $T=6.7+1=7.7$ .

### See Also

Event-Based Entity Generator (Obsolete), Entity Sink

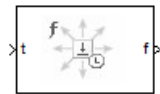
Introduced before R2006a

# Time-Based Function-Call Generator (Obsolete)

Generate function-call events in a time-based manner.

## Library

Generators/Function-Call Generators



## Description Time-Based Function-Call Generator

This block generates function-call events, either once at the start of simulation, using an intergeneration period that you specify in the block dialog box, or using a signal connected to the input port. The intergeneration period is the time interval between two successive generation events.

You can set the **Event generation mode** parameter of the block to one of three values. The block determines the intergeneration period differently for each value of the **Event generation mode** parameter that you choose.

Value of Event generation mode Parameter	Intergeneration Period
Only at simulation start	None, because only one event is generated
Period from dialog	Specified in the <b>Period</b> parameter of the block dialog box
Period from port	Specified using an input signal <b>t</b> that the block reads at the start of the simulation and each time it generates an entity

If you select the configuration parameter **Prevent duplicate events on multiport blocks and branched signals** in your model, the Time-Based Function-Call Generator block is compatible with all other blocks from SimEvents version 4.0 (R2011b), or later.

## Ports

### Signal Input Ports

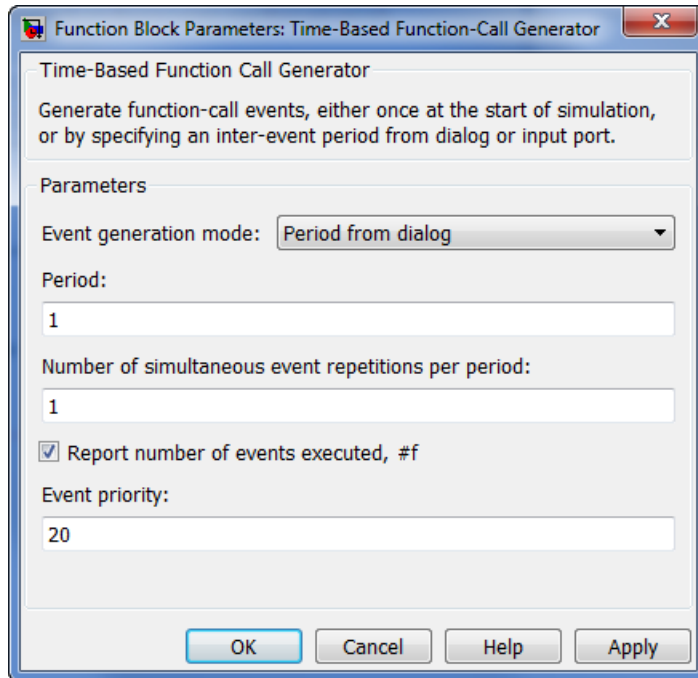
Label	Description
<b>t</b>	Time interval between generation of the current function-call event and the next function-call event. You see this port only if you set the <b>Event generation mode</b> parameter to <b>Period from port</b> in the block dialog box. At the start of the simulation and each time the block updates its output signals, the block reads the value at the input port, or if the preceding block is an <b>Event-Based Random Number</b> or <b>Event-Based Sequence</b> block, actively requests an updated input value. The signal connected to the input port must be an event-based signal.

### Signal Output Ports

Label	Description
<b>f</b>	Port through which generated function-call signals depart.
<b>#f</b>	Number of function-call events that have been executed by the block. You only see this port if you select the <b>Report number of events executed, #f</b> check box in the block dialog box.

The initial output value — in effect from the start of the simulation until the first update by the block— is 0, for all signals.

## Dialog Box



### Event generation mode

Determines the mode that the block uses to generate function-call events.

### Period

The time interval between generation of successive function-call events, in seconds. You see this field only if you set the **Event generation mode** parameter to **Period from dialog** in the block dialog box.

### Number of simultaneous event repetitions per period

The number of simultaneous function-call events that the block generates in each period. Use this parameter to generate a function-call with multiple iterations.

### Report number of events executed, #f

Determines if the number of function-call events that have executed by the block is made available via a signal output port on the block.

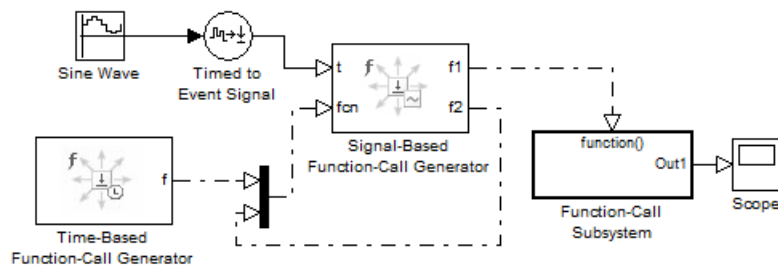
## Event priority

The priority of the function-call event relative to other simultaneous events in the simulation.

## Examples

### Seed Event Generation

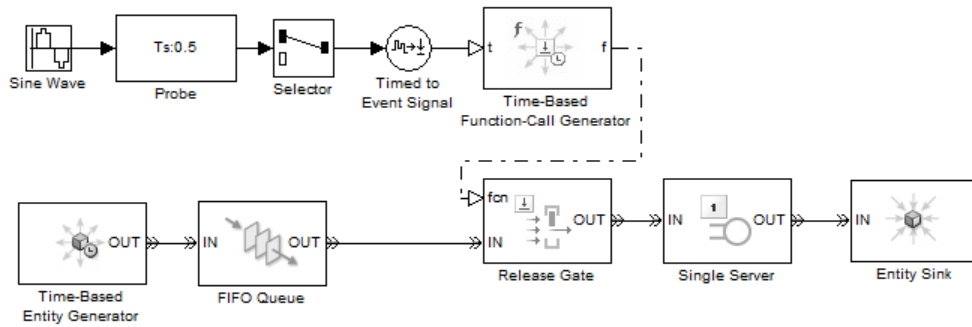
To generate a seed event in your model at simulation time  $T=0$ , you can use the **Time-Based Function-Call Generator** block. A seed event is an initial impulse that models with certain block configurations require to update the outputs of their blocks and to start generating events.



In this example, the **Event generation mode** parameter of the **Time-Based Function-Call Generator** block is set to **Only** at simulation start. At simulation time  $T=0$ , the **Time-Based Function-Call Generator** block produces an initial function-call event — or seed event — to the **Signal-Based Function-Call Generator** block. This seed event causes the **Signal-Based Function-Call Generator** block to update its outputs. After simulation time  $T=0$ , the simulation continues to update the outputs of the **Signal-Based Function-Call Generator** block. The model is now self-sustaining.

### Time Synchronization

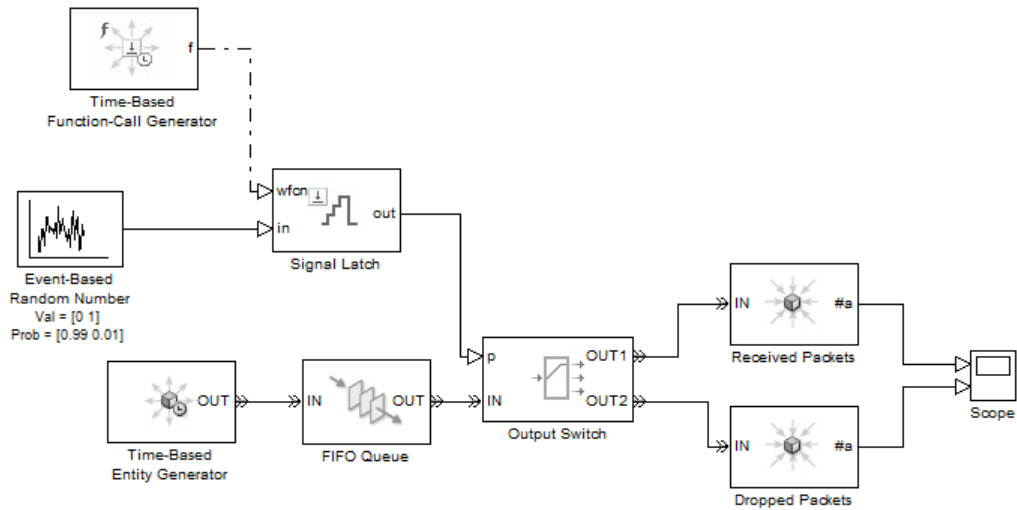
When you want to synchronize events with a time-based process outside the SimEvents domain, use the **Time-Based Function-Call Generator** block.



In this model, the **Event generation mode** parameter of the **Time-Based Function-Call Generator** block is set to **Period** from port. The **Probe** block detects the sample time of a sine wave signal and connects it to the input port **t** of the **Time-Based Function-Call Generator** block. The value at the input port **t** determines the period — or time delay — between successive function-call events generated by the block and is used to control entity advancement through the **Release Gate** block.

## Discrete-Event Model with Statistical Distribution

You can use the **Time-Based Function-Call Generator** block to model a discrete-event system that is driven by a time-based process drawn from a statistical distribution.



In this model, using a statistical distribution, the **Event-Based Random Number** block generates random values. The **Time-Based Generator** block periodically produces a function-call event to store the current output value of the **Event-Based Random Number** block in the **Signal Latch** block. This value is used, in turn, to select the output port of the **Output Switch** block. Based on the statistical distribution specified in this model, the value that is output by the **Signal Latch** block has a 99% probability of being **1**. When this situation is the case, the *first* output port of the **Output Switch** block is selected, and an entity advances as a received packet. Conversely, 1% of entities advance as dropped packets.

**Introduced in R2012a**



# Timed to Event Function-Call (Obsolete)

Convert time-based function call to event-based function call

## Library

Gateways

## Description

This block converts a scalar time-based function call into an event-based function call. The output signal is almost identical to the input signal, except that the output can be an input to a block that requires an event-based function-call input signal.

## Ports

### Signal Input Ports

Label	Description
None	Time-based function-call signal.

### Signal Output Ports

Label	Description
None	Event-based function-call signal.

## See Also

Event to Timed Function-Call (Obsolete)

“Time-Based Signals and SimEvents Block Transitions”

Introduced in R2011b

## Timed to Event Signal (Obsolete)

Convert time-based signal to event-based signal

### Library

Gateways

### Description

This block converts a time-based data signal into an event-based data signal. The value of the output signal is identical to that of the input signal. The output signal can be an input to a block that requires an event-based input signal.

### Ports

#### Signal Input Ports

Label	Description
None	Time-based signal. The signal can have any fixed dimension, complexity, or data type.

#### Signal Output Ports

Label	Description
None	Event-based signal

The initial output value is the same as the initial input value.

### See Also

Event to Timed Signal (Obsolete)

“Time-Based Signals and SimEvents Block Transitions”

**Introduced in R2011b**

## X-Y Attribute Scope (Obsolete)

Plot data from two attributes of arriving entities

### Library



### Description

This block plots a curve using data from two real scalar-valued attributes of arriving entities. Use the **Y attribute name** and **X attribute name** parameters to specify which attributes to plot.

Use the **Enable entity OUT port** option to choose whether the entity advances to a subsequent block or whether the block absorbs the arriving entity.

The **Plot type** parameter on the **Plotting** determines whether and how the block connects the points that it plots.

### Ports

#### Entity Input Ports

Label	Description
IN	Port for arriving entities, whose attributes contain the data to plot.

#### Entity Output Ports

Label	Description
OUT	Port for departing entities. You see this port only if you select <b>Enable entity OUT port</b> .

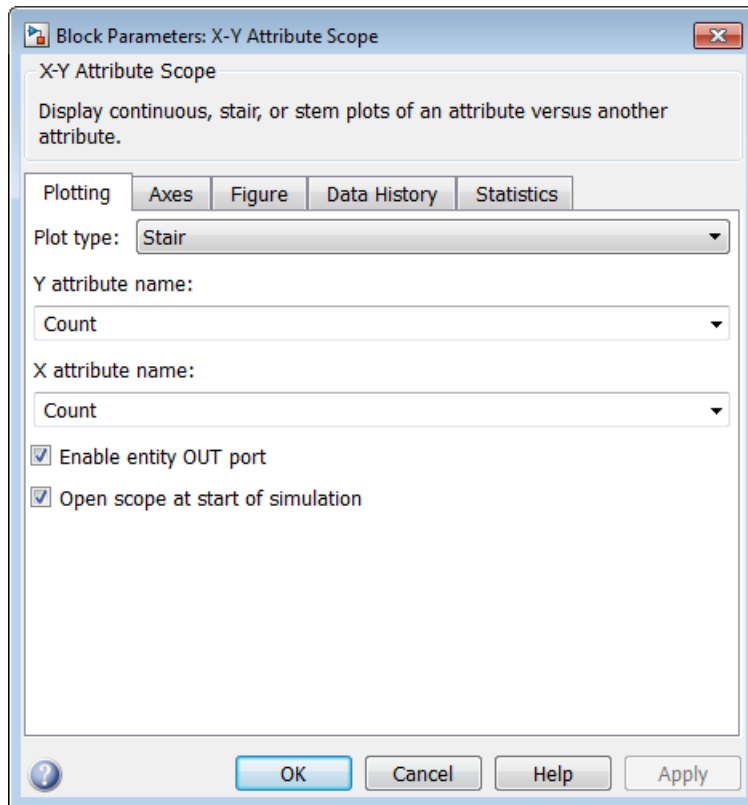
#### Signal Output Ports

Label	Description
#a	Number of entities that have arrived at the block since the start of the simulation.

## Dialog Box

To open the block dialog box, click the Parameters toolbar button in the plot window.

## Plotting



### Plot type

The presentation format for the data.

### Y attribute name

Name of the attribute to plot along the vertical axis.

### **X attribute name**

Name of the attribute to plot along the horizontal axis.

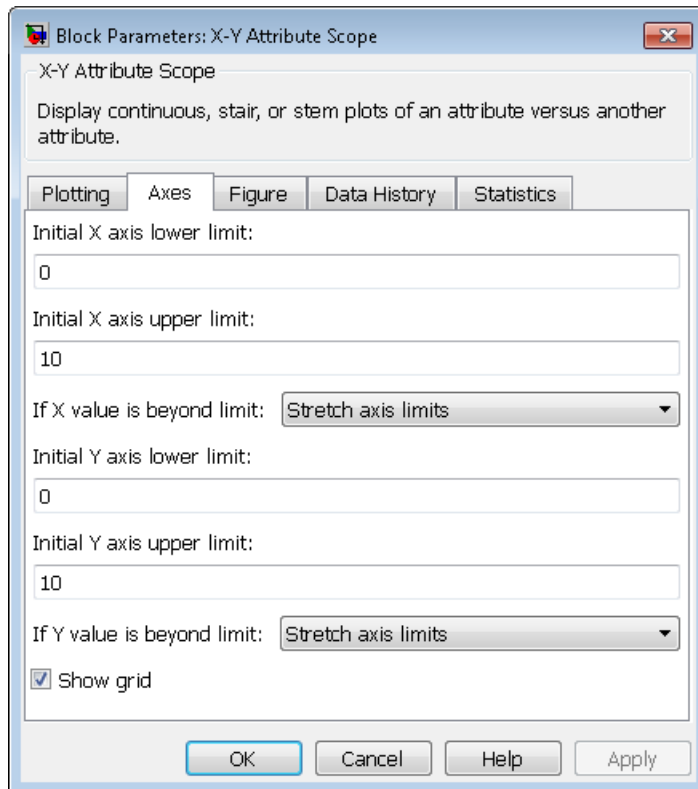
### **Enable entity OUT port**

Causes the block to have an entity output port labeled **OUT**, through which the arriving entity departs. If you clear this box, the block absorbs arriving entities.

### **Open scope at start of simulation**

Selecting this option causes the plot window to open when you start the simulation. If you clear this box, you can open the plot window by double-clicking the block icon.

## **Axes**



### **Initial X axis lower limit, Initial X axis upper limit**

The interval shown on the X axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If X value is beyond limit** setting.

**If X value is beyond limit**

Determines how the plot changes if one or more X values are not within the limits shown on the X axis.

**Initial Y axis lower limit, Initial Y axis upper limit**

The interval shown on the Y axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If Y value is beyond limit** setting.

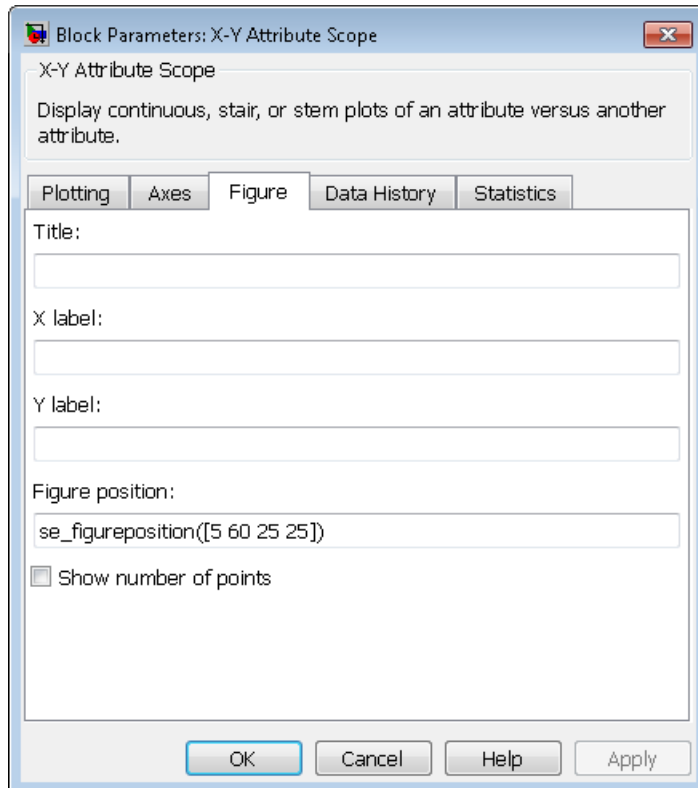
**If Y value is beyond limit**

Determines how the plot changes if one or more Y values are not within the limits shown on the Y axis.

**Show grid**

Toggles the grid on and off.

### Figure



#### Title

Text that appears as the title of the plot, above the axes.

#### Y label

Text that appears to the left of the vertical axis.

#### X label

Text that appears below the horizontal axis.

#### Figure Position

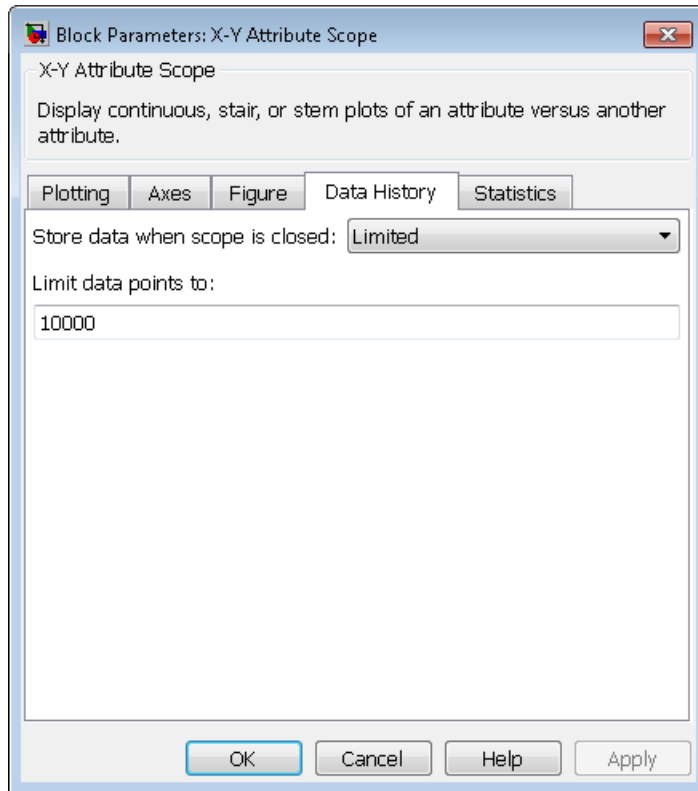
A four-element vector of the form [left bottom width height] specifying the position of the scope window. (0,0) is the lower left corner of the display.



### Show number of entities

Displays the number of plotted points using an annotation in the plot window.

## Data History



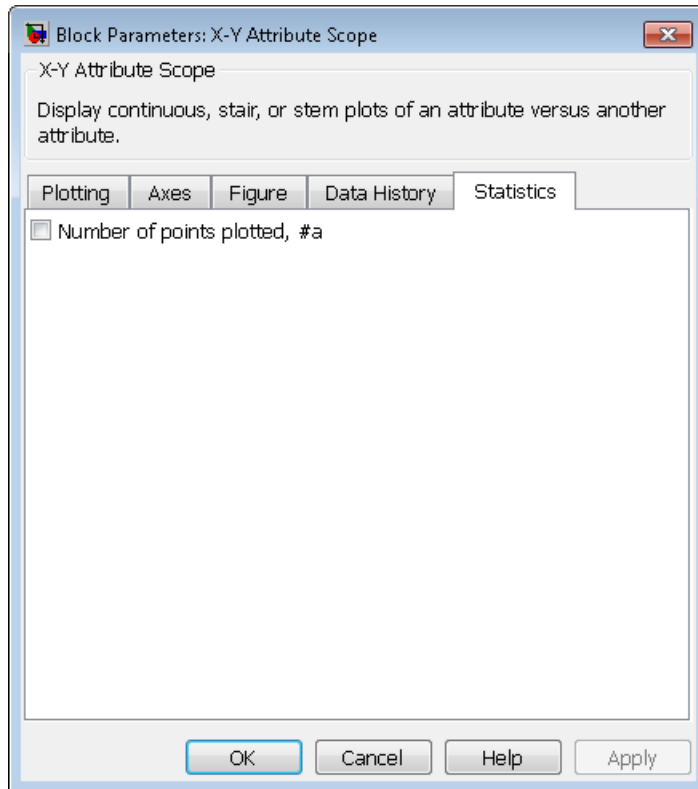
### Store data when scope is closed

Select **Unlimited** to have the block cache all data for future viewing, **Limited** to cache a portion of the most recent data, and **Disabled** to avoid caching undisplayed data.

### Limit data points to

The number of data points the block caches, using the most recent data. **Store data when scope is closed** to **Limited**.

## Statistics



### Number of points plotted

#a.

## Examples

This section is not available for prerelease.

## See Also

Attribute Scope (Obsolete), X-Y Signal Scope (Obsolete)

“Manipulate Entity Attributes”

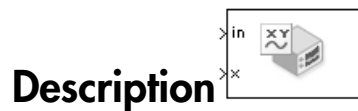
**Introduced before R2006a**

## X-Y Signal Scope (Obsolete)

Plot data from two signals

### Library

SimEvents Sinks



### Description

The **Plot type** parameter on the **Plotting** tab determines whether and how the block connects the points that it plots.

### Ports

#### Signal Input Ports

Label	Description
in	Signal containing data for Y axis. This signal must be an event-based signal.
x	Signal containing data for X axis. This signal must be an event-based signal.

#### Signal Output Ports

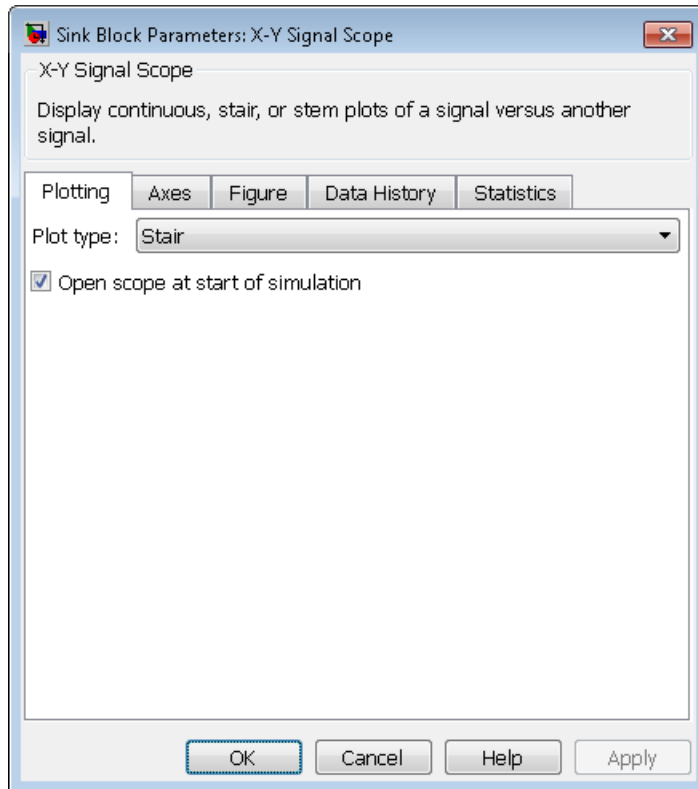
Label	Description
#c	Number of points the block has plotted.

The initial output value is 0. This value is in effect from the start of the simulation until the first update by the block.

### Dialog Box

To open the block dialog box, click the Parameters toolbar button in the plot window.

## Plotting Tab



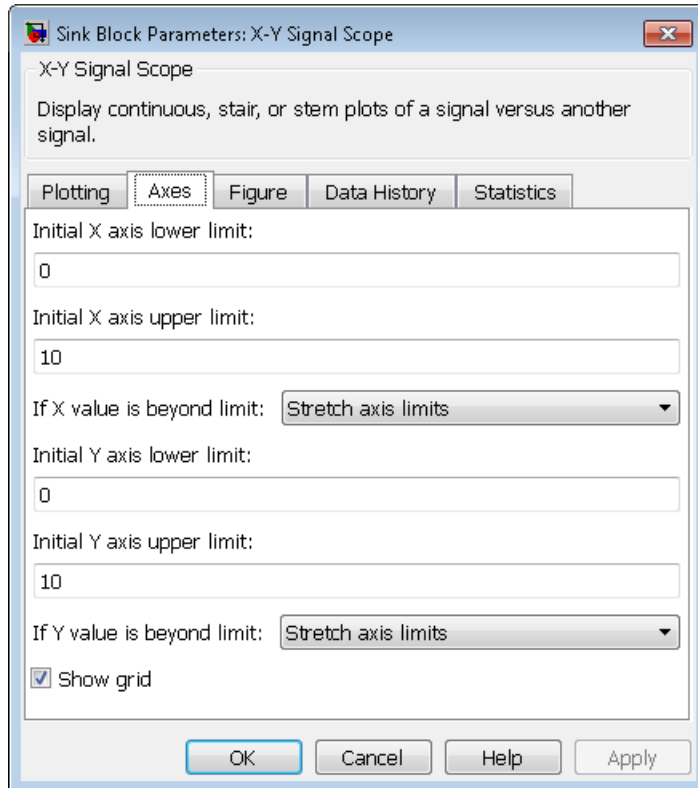
### Plot type

The presentation format for the data.

### Open scope at start of simulation

Selecting this option causes the plot window to open when you start the simulation. If you clear this box, you can open the plot window by double-clicking the block icon.

## Axes Tab



### Initial X axis lower limit, Initial X axis upper limit

The interval shown on the X axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If X value is beyond limit** setting.

### If X value is beyond limit

Determines how the plot changes if one or more X values are not within the limits shown on the X axis.

### Initial Y axis lower limit, Initial Y axis upper limit

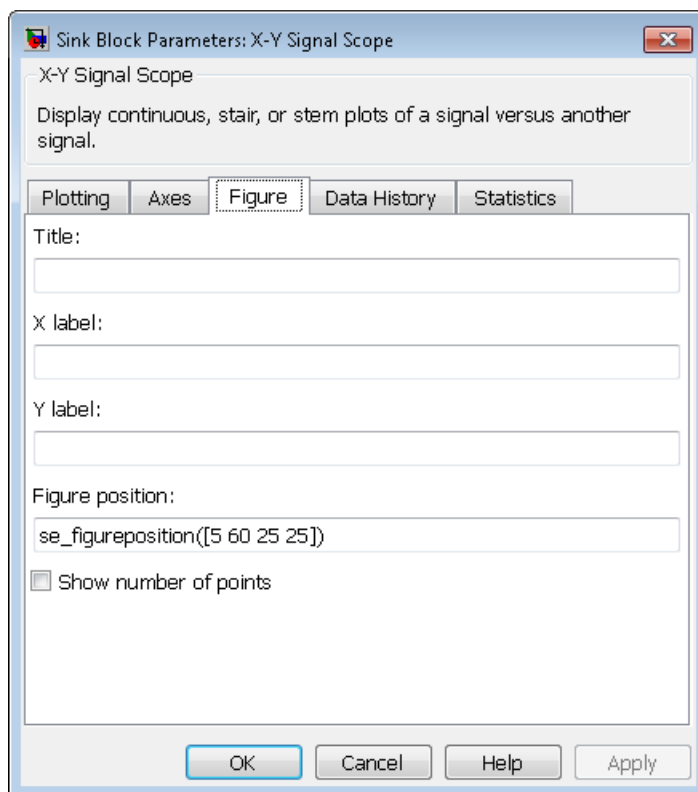
The interval shown on the Y axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If Y value is beyond limit** setting.

**If Y value is beyond limit**

Determines how the plot changes if one or more Y values are not within the limits shown on the Y axis.

**Show grid**

Toggles the grid on and off.

**Figure Tab****Title**

Text that appears as the title of the plot, above the axes.

**Y label**

Text that appears to the left of the vertical axis.

### X label

Text that appears below the horizontal axis.

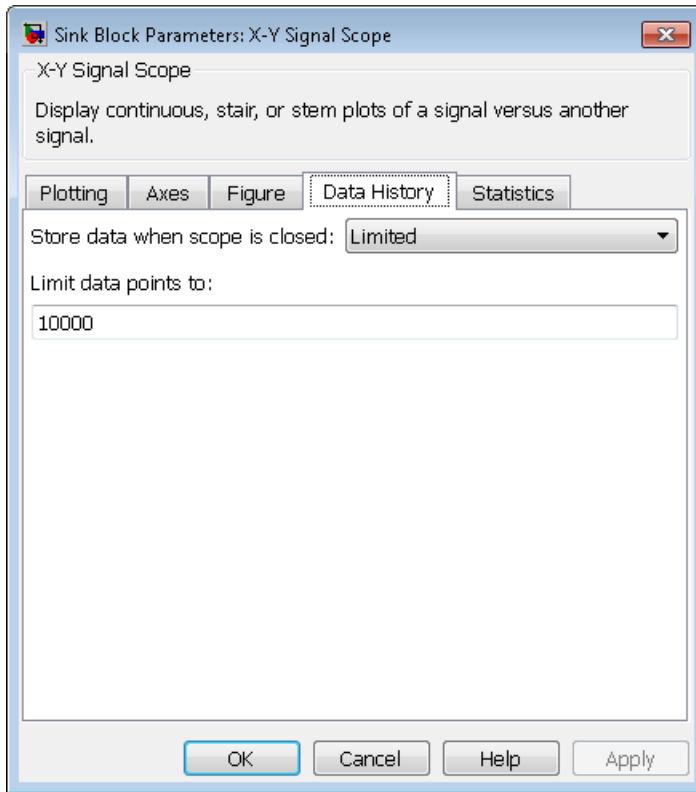
### Figure Position

A four-element vector of the form [left bottom width height] specifying the position of the scope window. (0,0) is the lower left corner of the display.

### Show number of points

Displays the number of plotted points using an annotation in the plot window.

## Data History Tab



### Store data when scope is closed



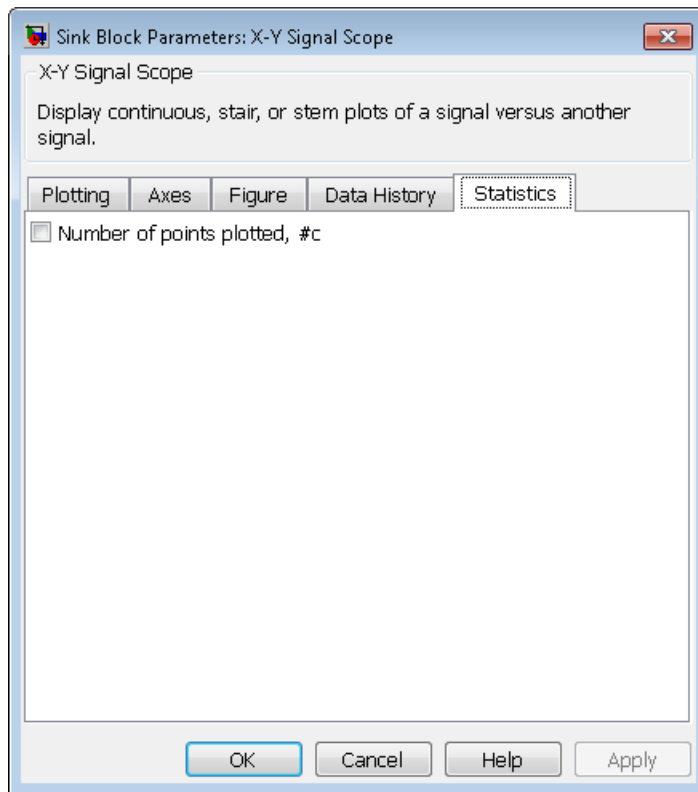
Select **Unlimited** to have the block cache all data for future viewing, **Limited** to cache a portion of the most recent data, and **Disabled** to avoid caching undisplayed data.

### Limit data points to

The number of data points the block caches, using the most recent data. You see this field only if you set **Store data when scope is closed** to **Limited**.

## Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see the preceding table, “Signal Output Ports”.

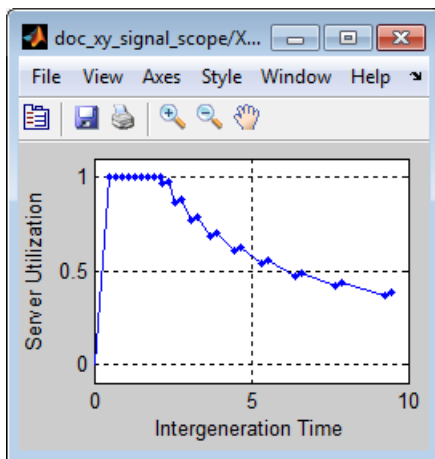
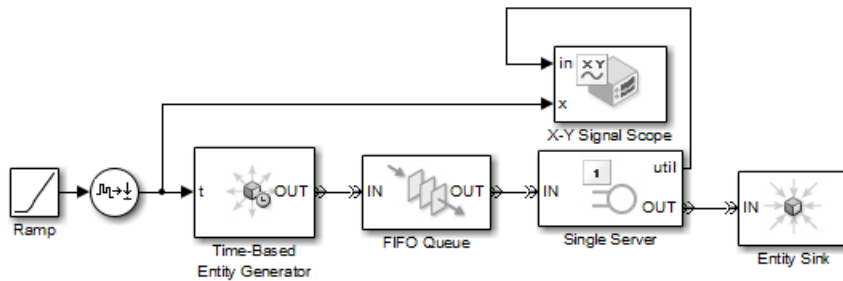


**Number of points plotted, #c**

Allows you to use the signal output port labeled #c.

## Examples

The model below shows the relationship between the utilization of a server and the interarrival time of entities.



## See Also

Signal Scope (Obsolete), X-Y Attribute Scope (Obsolete)

**Introduced before R2006a**



# Configuration Parameters

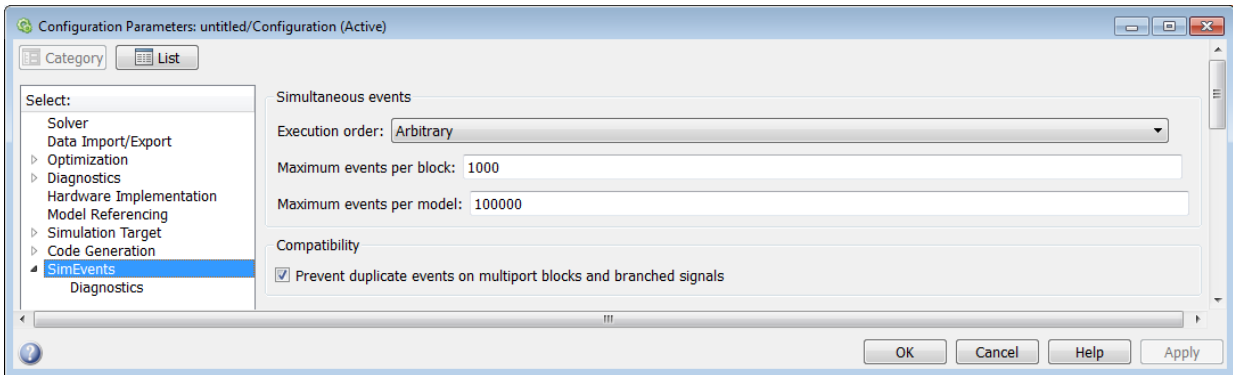
---

- “SimEvents Pane” on page 3-2
- “SimEvents Diagnostics Pane” on page 3-10

## SimEvents Pane

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---



**In this section...**

“SimEvents Pane Overview” on page 3-4

“Execution order” on page 3-5

“Seed for event randomization” on page 3-6

“Maximum events per block” on page 3-7

“Maximum events per model” on page 3-8

“Prevent duplicate events on multiport blocks and branched signals” on page 3-8

### SimEvents Pane Overview

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Configure modelwide parameters related to discrete-event simulation and the logging of events and entities.

#### Configuration

This pane appears only if your model contains a SimEvents block.



## Execution order

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Select an algorithm for determining the sequence for processing simultaneous events having equal priorities.

### Settings

**Default:** Arbitrary

#### Arbitrary

Causes the simulation to use an internal algorithm to determine the sequence for processing simultaneous events having equal priorities.

#### Randomized

Causes the simulation to assign equal probability to all possible execution sequences of simultaneous events having equal numerical priorities.

### Tip

The processing sequence might be different from the sequence in which the events were scheduled on the event calendar.

### Dependency

Selecting Randomized enables **Seed for event randomization**.

### Command-Line Information

**Parameter:** propIdentEvents

**Type:** double

**Value:** 0 | 1

**Default:** 0

### Seed for event randomization

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Initialize the random number generator for event processing.

#### Settings

**Default:** 123456789

**Minimum:** 0

**Maximum:**  $2^{31} - 1$

This is a number that initializes the random number generator used to determine the sequence for processing simultaneous events having equal priorities.

#### Tips

- For a given value of this parameter, the output of the random number generator is repeatable.
- To avoid unexpected correlations, make the value of this parameter distinct from all other seed parameters in the model (for example, the **Initial seed** parameter in the Event-Based Random Number block).

#### Dependency

This parameter is enabled by **Execution order**.

#### Command-Line Information

**Parameter:** propIdentEventSeed

**Type:** string

**Value:**

**Default:** '123456789'

## Maximum events per block

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Limit the number of entity generation, service completion, subsystem execution, and function-call events that each SimEvents block performs at each fixed time instant.

### Settings

**Default:** 1000

**Minimum:** 2

**Maximum:**  $2^{31} - 1$

### Command-Line Information

**Parameter:** propMaxDesBlkSimulEvents

**Type:** string

**Value:**

**Default:** '1000'

## Maximum events per model

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Limit the total number of events scheduled via the event calendar at each fixed time instant. This is the maximum number of events per discrete-event system in a model.

### Settings

**Default:** 100000

**Minimum:** 2

**Maximum:**  $2^{31} - 1$

### Command-Line Information

**Parameter:** propMaxDesMdlSimulEvents

**Type:** string

**Value:**

**Default:** '100000'

## Prevent duplicate events on multiport blocks and branched signals

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Prevent multifiring behavior on multiport blocks or branched signals that results in duplication of events. Multifiring behavior, an implicit result of the way that the software executes particular block configurations, occurs when the software executes a block more than once in response to a single, discrete event in the simulation.

### Settings

**Default:** On

On

Enable **Prevent duplicate events on multiport blocks and branched signals** parameter to prevent multifiring behavior.

Off

Allow multifiring behavior on multiport blocks or branched signals.

#### **Command-Line Information**

**Parameter:** propPreventDuplicateEvents

**Type:** integer or boolean

**Value:**

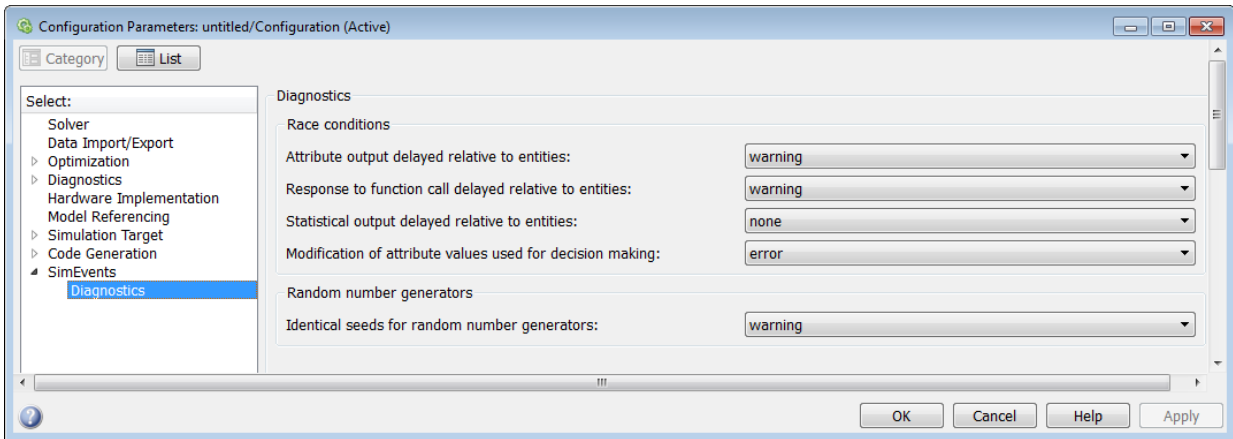
**Default:** '1' for integer, 'True' for boolean

#### **More About**

- “SimEvents Diagnostics Pane” on page 3-10

## SimEvents Diagnostics Pane

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.



**In this section...**

“Diagnostics Pane Overview” on page 3-12

“Attribute output delayed relative to entities” on page 3-13

“Response to function call delayed relative to entities” on page 3-15

“Statistical output delayed relative to entities” on page 3-17

“Modification of attribute values used for decision making” on page 3-19

“Identical seeds for random number generators” on page 3-21

### Diagnostics Pane Overview

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Specify what diagnostic action the application should take, if any, when it detects situations that might cause problems or unexpected results in the simulation.

#### Configuration

This pane appears only if your model contains a SimEvents block.

#### Tips

- The options are typically to do nothing or to display a warning or an error message.
- A warning does not terminate a simulation, but an error does.



## Attribute output delayed relative to entities

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Select the diagnostic action to take if the application detects a situation in which a **Get Attribute** block updates a signal during entity advancement, but a subsequent block responds to the signal update after the entity has arrived. The application's processing sequence might cause the latter block to process the entity using outdated signal values.

### Settings

#### Default: error

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

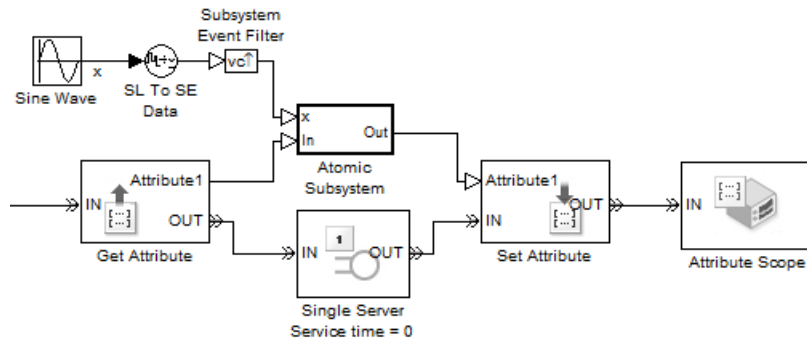
error

When the application detects this situation, it terminates the simulation and displays an error message.

### Tip

A **Single Server** block whose **Service time** parameter is 0 can address the problem by storing the entity while the latter block responds to the signal update.

#### Example of Solution



Alternatively, you might be able to address the problem by using an attribute directly instead of by using the signal output of a Get Attribute block.

#### Command-Line Information

**Parameter:** propDiagAttribOutput

**Type:** double

**Value:** 0 | 1 | 2

**Default:** 2

#### Recommended Settings

Application	Setting
Debugging	warning or error
Efficiency	none

## Response to function call delayed relative to entities

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Select the diagnostic action to take if the application detects a situation in which a block issues a function call during entity advancement, but subsequent blocks respond to the function call and its consequences after the entity has arrived. The application's processing sequence might cause subsequent blocks to process the entity using outdated values of a signal whose update is a consequence of the function call.

### Settings

#### Default: error

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

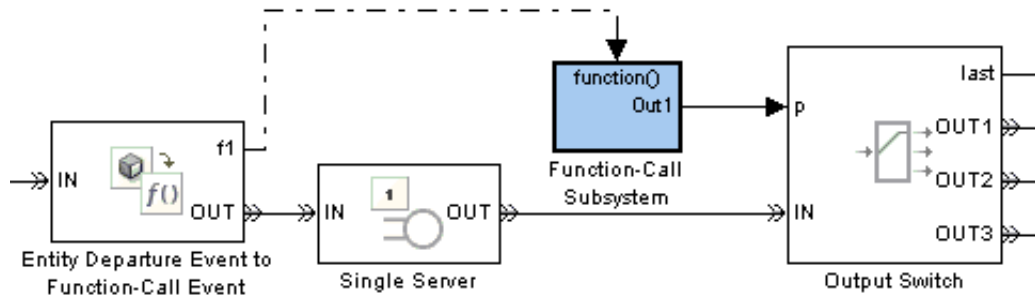
error

When the application detects this situation, it terminates the simulation and displays an error message.

### Tip

A **Single Server** block whose **Service time** parameter is 0 can address the problem by storing the entity while subsequent blocks respond to the function call and its consequences.

**Example of Solution**



**Command-Line Information**

**Parameter:** propDiagFcnCallOutput

**Type:** double

**Value:** 0 | 1 | 2

**Default:** 2

**Recommended Settings**

Application	Setting
Debugging	warning or error
Efficiency	none

## Statistical output delayed relative to entities

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Select the diagnostic action to take if the application detects a situation in which a block updates a statistical output signal during entity advancement, but a subsequent block responds to the signal update after the entity has arrived. The application's processing sequence might cause the latter block to process the entity using outdated signal values.

### Settings

**Default:** error

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

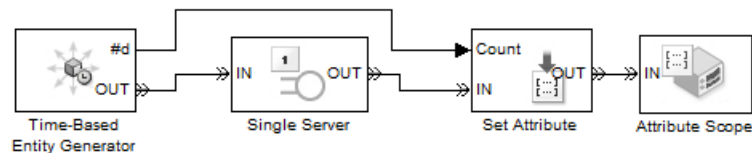
error

When the application detects this situation, it terminates the simulation and displays an error message.

### Tip

A **Single Server** block whose **Service time** parameter is 0 can address the problem by storing the entity while the latter block responds to the signal update.

### Example of Solution



### Command-Line Information

**Parameter:** propDiagStatOutput

**Type:** double

**Value:** 0 | 1 | 2

**Default:** 1

#### **Recommended Settings**

<b>Application</b>	<b>Setting</b>
Debugging	warning or error
Efficiency	none

## Modification of attribute values used for decision making

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Select the diagnostic action to take if the application detects certain situations in which a block modifies an attribute that a subsequent block uses to determine its availability. In some of these cases, internal queries among blocks might result in a decision based on information that changes when the entity actually advances.

### Settings

#### Default: error

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

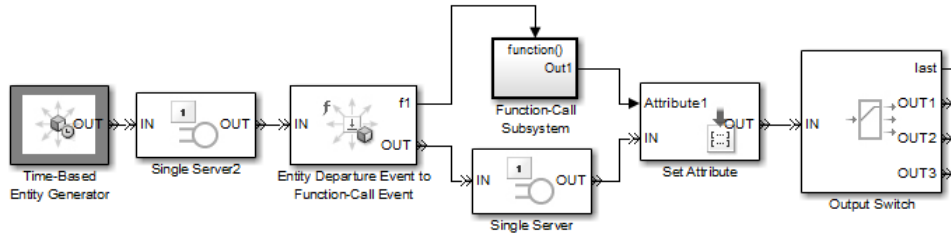
error

When the application detects this situation, it terminates the simulation and displays an error message.

### Tip

A **Single Server** block whose **Service time** parameter is 0 can address the problem by storing the entity while the latter block responds to the signal update.

#### Example of Solution



#### Command-Line Information

Parameter: propDiagChangeAttrib

Type: double

Value: 0 | 1 | 2

Default: 2

#### Recommended Settings

Application	Setting
Debugging	warning or error
Efficiency	none



## Identical seeds for random number generators

---

**Note:** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Select the diagnostic action to take if the application detects that multiple random number generators use the same seed value, which might cause correlations among random processes.

### Settings

**Default:** warning

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

error

When the application detects this situation, it terminates the simulation and displays an error message.

### Tips

- If you set the parameter to **warning**, the warning message contains hyperlinks labeled “Randomize” and “Randomize All” that can help you address the problem.
- The `se_randomize_seeds` function provides a programmatic way to address the problem.
- Set the parameter to **none** if duplicate seeds are intentional in your model.

### Command-Line Information

**Parameter:** `propRNGIdenticalSeeds`

**Type:** double

**Value:** 0 | 1 | 2

**Default:** 1

### Recommended Settings

Application	Setting
Debugging	warning or error
Efficiency	none

### More About

- “SimEvents Pane” on page 3-2

# Upgrade Advisor Checks

---

# SimEvents Upgrade Advisor Checks

---

**Note:** These checks are obsolete. They are available only for SimEvents releases prior to R2016a.

---

In this section...
“Checks Overview” on page 4-2
“Check for implicit event duplication caused by SimEvents blocks” on page 4-3

## Checks Overview

---

**Note:** These checks are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Use SimEvents Upgrade Advisor checks to identify backward-compatibility issues in your model.

## Check for implicit event duplication caused by SimEvents blocks

---

**Note:** These checks are obsolete. They are available only for SimEvents releases prior to R2016a.

---

Check configuration parameters of model for status of **Prevent duplicate events on multiport blocks and branched signals** option.

### Description

This Upgrade Advisor check verifies if you have selected the **Prevent duplicate events on multiport blocks and branched signals** check box in the Configuration Parameters dialog box of your model.

When you run a model created in a version of SimEvents prior to R2012a, the model might exhibit a behavior called *multifiring* that leads to duplication of events in the simulation. This event duplication behavior is implicit in models with certain configurations and results from the way the software executes the blocks of such configurations. Implicit event duplication is resolved in R2012a with the addition of the configuration parameter **Prevent duplicate events on multiport blocks and branched signals**.

Available with SimEvents.

### Results and Recommended Actions

Condition	Recommended Action
SimEvents > <b>Prevent duplicate events on multiport blocks and branched signals</b> check box is not selected.	In the Configuration Parameters dialog box of your model, select the SimEvents > <b>Prevent duplicate events on multiport blocks and branched signals</b> check box.

An alternative to the recommended action in the preceding table is to use the **Modify Settings** button in the **Action** section of the Upgrade Advisor results pane. If you click **Modify Settings**, the software directly enables **Prevent duplicate events on multiport blocks and branched signals**.

---

**Note:** The configuration parameter **Prevent duplicate events on multiport blocks and branched signals** is not compatible with blocks from versions of SimEvents prior

to 4.0 (R2011b). The Upgrade Advisor provides the recommended action (if any) for the check, “Check for implicit event duplication caused by SimEvents blocks” on page 4-3.

---

### **More About**

- “Consult the Upgrade Advisor”

<b>advance</b>	To depart from one block and arrive immediately at another block. An entity advances from block to block during a simulation.
<b>arrival</b>	Entrance of an entity to a block via an entity input port. Arrival is the opposite of departure.
<b>attribute</b>	Data associated with an entity.  For example, an entity might be associated with a size, weight, speed, or part number.
<b>available</b>	The state of an entity input port that permits entities to arrive at the block.  For example, when a <b>Entity Server</b> block is empty, its entity input port is available. When the block is busy serving, its entity input port is unavailable.
<b>blocked</b>	The state of an entity output port when an entity is trying to depart via the port and the port connects to an unavailable entity input port of another block.  For example, consider a <b>Entity Queue</b> block whose entity output port is connected to the <b>Entity Server</b> block's entity input port. Suppose the queue contains one entity. The queue's entity output port is blocked if the server's entity input port is unavailable, and not blocked if the server's entity input port is available. If the queue is empty, then its entity output port is not blocked because no entity is trying to depart.
<b>component entity</b>	An entity that forms part of a composite entity.
<b>composite entity</b>	An entity that comprises one or more entities as subordinate parts. The parts are called component entities.
<b>departure</b>	Exit of an entity from a block via an entity output port. Departure is the opposite of arrival.

**discrete-event system**

A system in which state transitions depend on asynchronous discrete incidents called events. You typically construct a discrete-event system by adding a variety of blocks, such as generators, queues, and servers, from the SimEvents block library.

One or more discrete-event systems can coexist with time-based systems in a Simulink model. SimEvents software automatically handles signals transitioning from time-based components/systems to and from discrete-event components/systems and labels these signal lines with a capital **E**.

**entity**

An abstract representation of an item of interest in a discrete-event simulation. The specific interpretation of an entity depends on what you are modeling. Entities can carry data, known as attributes.

For example, an entity could represent a packet in a communication network, a person using a bank of elevators, or a part on a conveyor belt.

**entity input port**

An input port at which an entity can potentially arrive. An entity input port can be available or unavailable; this state, which can change during the simulation, helps determine whether the port actually accepts the arrivals of new entities.

**entity output port**

An output port from which an entity can potentially depart. An entity output port can have a state of blocked or not blocked; this state, which can change during the simulation, determines whether the port's attempt to output an entity is successful.

**entity path**

A connection from an entity output port to an entity input port, depicted as a line connecting the entity ports of two blocks. An entity path represents the equivalence between an entity's departure from the first block and arrival at the second block. The connection line depicts a relationship between the two blocks.



	<p>An entity path is in active use by an entity only at zero or more discrete times during the simulation. By contrast, a connection line between signal ports represents a signal that has a well-defined value at all times during the simulation.</p>
<b>entity port</b>	<p>An entity input port or an entity output port.</p>
<b>entity priority</b>	<p>A positive number associated with an entity, used to sequence its departure with regard to other simultaneous departures. A lower number indicates a higher priority.</p>
<b>event</b>	<p>An observation of an instantaneous incident that may change a state variable, an output, and/or the occurrence of other events. Examples of events are the generation of a new data packet in communications, the exit of a person from an elevator, and the placement of a new part on a conveyor belt.</p>
<b>event calendar</b>	<p>The internal list of events that are scheduled for the current time or future times.</p> <p>For example, when a server begins its service time on a specific entity, the application inserts an entry into the event calendar for the completion of service on that entity at a future time. In a system representing elevator passengers, this event calendar entry might represent the event whereby a specific person in an elevator reaches the desired floor.</p>
<b>event-based simulation</b>	<p>A simulation that permits the system's state transitions to depend on asynchronous discrete incidents called events.</p>
<b>intergeneration time</b>	<p>The time interval between successive generations.</p>
<b>pending entity</b>	<p>An entity that has tried and failed to depart from the block in which the entity resides. The failure occurs because the entity output port through which the entity would depart is connected to an unavailable entity input port of another block.</p>

<b>preemption</b>	The replacement of an entity in a server block by an entity that satisfies certain criteria.
<b>signal port</b>	An input or output port that represents a numerical quantity that changes over time and that is defined for all times during the simulation. Unlike an entity port, a signal port has no state and does not have entity arrivals or entity departures.
<b>simultaneous events</b>	<p>Events that occur at the same value, or sufficiently close values, of the simulation clock. Events scheduled on the event calendar for times <math>T</math> and <math>T+\Delta t</math> are considered simultaneous if <math>0 \leq \Delta t \leq 128 * \text{eps} * T</math>, where <b>eps</b> is the floating-point relative accuracy in MATLAB software and <math>T</math> is the simulation time.</p> <p>For example, in a D/D/1 queuing system where the arrival rate equals the service rate, an entity generation event and a service completion event are simultaneous. Parameters in the model determine which of these events occurs first, though the clock has the same value in both cases.</p>
<b>time-based signal</b>	A signal that can change only in response to the simulation clock.
<b>time-based simulation</b>	<p>A simulation in which state transitions depend on time.</p> <p>For example, a simulation based solely on differential equations in which time is an independent variable is a time-based simulation.</p>
<b>unavailable</b>	<p>The state of an entity input port that prevents entities from arriving at the block.</p> <p>For example, when an <b>Entity Server</b> block is empty, its entity input port is available. When the block is busy serving, its entity input port is unavailable.</p>
<b>zero-duration value</b>	A value that an event-based signal assumes at an instant in time but that does not persist for a positive duration.

For example, when a full N-server advances one entity to the next block, the statistical signal representing the number of entities in the block assumes the value N-1. However, if the departure causes another entity to arrive at the block at the same time instant, then the statistical signal assumes the value N. The value of N-1, which does not persist for a positive duration, is a zero-duration value.. This phenomenon occurs in many situations.

