SimEvents®
Reference

MATLAB&SIMULINK®

R2021**a**

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

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

Phone: 508-647-7000

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

SimEvents® Reference

© COPYRIGHT 2005–2021 by The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See 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 for more information.

Revision History

March 2007	Online only	Revised for Version 2.0 (Release 2007a). Previously part of *SimEvents® User's Guide*.
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)
September 2016	Online only	Revised for Version 5.1 (Release 2016b)
March 2017	Online only	Revised for Version 5.2 (Release 2017a)
September 2017	Online only	Revised for Version 5.3 (Release 2017b)
March 2018	Online only	Revised for Version 5.4 (Release 2018a)
September 2018	Online only	Revised for Version 5.5 (Release 2018b)
March 2019	Online only	Revised for Version 5.6 (Release 2019a)
September 2019	Online only	Revised for Version 5.7 (Release 2019b)
March 2020	Online only	Revised for Version 5.8 (Release 2020a)
September 2020	Online only	Revised for Version 5.9 (Release 2020b)
March 2021	Online only	Revised for Version 5.10 (Release 2021a)

Contents

Functions

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. For more information about implementing matlab.DiscreteEventSystem class with MATLAB Discrete-Event System block, see "Create Custom Blocks Using MATLAB Discrete-Event System Block".

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

For more information about implementing a discrete-event System object™, see "Create a Discrete-Event System Object". For information about linking the discrete-event System object to a SimEvents model and creating a custom behavior, see "Delay Entities with a Custom Entity Storage Block".

The matlab.DiscreteEventSystem class is a handle class.

Class Attributes

```
Abstract                                    false
HandleCompatible                            true
StrictDefaults                              false
```

For information on class attributes, see "Class Attributes".

Methods

Public Methods

entityType	Define entity type
blocked	Event action when entity forward fails
destroy	Event action upon entity destruction
entry	Event action when entity enters storage element
exit	Event action before entity exit from storage
generate	Event action upon entity creation
iterate	Event action when entity iterates
modified	Event action upon entity modification by the Entity Find block
resourceAcquired	Event action upon successful resource acquisition
resourceReleased	Event action upon successful resource release
testEntry	Event action to accept or refuse entity
timer	Event action when timer completes
setupEvents	Initialize entity generation events

queueFIFO	Define first-in first-out (FIFO) queue storage
queueLIFO	Define last-in last-out (LIFO) queue storage
queuePriority	Define priority queue storage
queueSysPriority	Define system priority queue storage
resourceSpecification	Create specifications for a resource acquisition or a resource release event
resourceType	Specify an entity type and the name of the resources to be acquired by the specified entity
initResourceArray	Initialize a resource specification array
eventAcquireResource	Create a resource acquisition event
eventDestroy	Create entity destroy event
eventForward	Create entity forward event
eventGenerate	Create entity generate event
eventIterate	Create entity iterate event
eventReleaseResource	Create an event to release previously acquired resources
eventReleaseAllResources	Create an event to release all resources acquired by an entity
eventTestEntry	Create an event to indicate that the acceptance policy for the storage has changed and the storage retests arriving entities
eventTimer	Create entity timer event
cancelAcquireResource	Cancel previously scheduled resource acquisition event
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

Protected Methods

initEventArray	Initialize event array
getEntityPortsImpl	Define input ports and output ports of discrete-event system
getEntityStorageImpl	Define entity storage elements of discrete-event system
getEntityTypesImpl	Define entity types of discrete-event system
getResourceNamesImpl	Define resource pools from which to acquire resources

Examples

Create a Custom Entity Storage Block to Delay Entities

This example shows how to use discrete-event System object methods to create a custom entity storage block that has one input port, one output port, and one storage element. The discrete-event System object is the instantiation of the `matlab.DiscreteEventSystem` class, which allows you to use the implementation and service methods provided by this class. Then, you use the MATLAB Discrete-Event System block to integrate the System object into a SimEvents model. The custom MATLAB Discrete-Event System block accepts an entity from its input port and forwards it to its output port with a specified delay. For more information, see "Delay Entities with a Custom Entity Storage Block".

```
classdef CustomEntityStorageBlock < matlab.DiscreteEventSystem

    % A custom entity storage block with one input, one output, and one storage.

    % Nontunable properties
    properties (Nontunable)
    % Capacity
        Capacity = 1;
```

```
    % Delay
        Delay=4;
    end

    methods (Access=protected)
        function num = getNumInputsImpl(~)
            num = 1;
        end

        function num = getNumOutputsImpl(~)
            num = 1;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityTypes = obj.entityType('Car');
        end

        function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
            inputTypes = {'Car'};
            outputTypes = {'Car'};
        end

        function [storageSpecs, I, O] = getEntityStorageImpl(obj)
            storageSpecs = obj.queueFIFO('Car', obj.Capacity);
            I = 1;
            O = 1;
        end

    end

    methods

        function [entity,event] = CarEntry(obj,storage,entity,source)
            % Specify event actions when entity enters storage.

             event = obj.eventForward('output', 1, obj.Delay);

        end

    end

end
```

See Also
entityType | entry | eventForward | eventGenerate | getEntityStorageImpl | matlab.System | queueFIFO

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"
"Create a Custom Entity Storage Block with Iteration Event"
"Create a Discrete-Event System Object"
"Custom Entity Storage Block with Multiple Timer Events"
Class Attributes
Property Attributes

Introduced in R2016a

blocked

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Event action when entity forward fails

Syntax

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

Description

`[entity,events]=blocked(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,...]=blocked(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. Entity has these fields:

- `sys` (MATLAB `structure`) — It has these fields:

  - `id` (double) — Entity ID
  - `priority` (double) — Entity priority
- `data` — Entity data

`destination` — Destination
MATLAB structure

Destination of entity, such as an output port or a storage element. It has these fields:

- type (character vector) — Specify output or storage
- index (double) — Output or storage index

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. Use matlab.DiscreteEventSystem class methods to create events. Each event has these fields:

- type (character vector) — Type of the event
- delay (double) — Delay before the event
- priority (double) — Priority of the event
- Storage (double) — Index of the storage element
- tag (character vector) — Event tag
- location (MATLAB structure) — Source or destination location of entity

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] = blocked(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
```

See Also

destroy | entry | exit | generate | getEntityPortsImpl | getEntityStorageImpl | getEntityTypesImpl | iterate | matlab.DiscreteEventSystem | setupEvents | timer

Topics

"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

cancelAcquireResource

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Cancel previously scheduled resource acquisition event

Syntax

`event = cancelAcquireResource(tag)`

Description

`event = cancelAcquireResource(tag)` cancels a previously scheduled resource acquisition event.

Input Arguments

`tag` — Tag of the previously scheduled resource acquisition event
character vector

Tag of the previously scheduled resource acquisition event to be canceled.

Output Arguments

`event` — Cancellation Event
MATLAB structure

Event for canceling the previously scheduled resource acquisition.

Examples

Cancel a Resource Acquisition Event

Cancel a resource acquisition event when the previously scheduled resource acquisition event times out.

```
function [entity,events] = timer(obj,storage,entity,tag)
    % Assume that eventTimer() defines a timer with tag and that there is a previously
    % scheduled resource acquisition event with 'loadingWorker' tag.
    event = obj.cancelAcquireResource('loadingWorker');
    % This cancels the 'loadingWorker' acquisition event when the timer finishes.
end
```

See Also

`eventForward` | `eventReleaseResource` | `getResourceNamesImpl` | `matlab.DiscreteEventSystem` | `resourceReleased`

Topics
"Create the Discrete-Event System Object with Multiple Timer Events"

"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2019a

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 `entry` or `exit`.

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] = timer(obj,storage,entity,tag)
    % Cancel the previously scheduled destroy event of the entity in
    % current event action context.
    event = obj.cancelDestroy();
end
```

See Also
cancelForward | cancelGenerate | cancelIterate | cancelTimer | eventDestroy | eventForward | eventGenerate | eventIterate | eventTimer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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 `entry` or `exit`.

Input Arguments

`destinationType` — Destination type
character vector

Destination type. 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] = timer(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
```

See Also

cancelDestroy | cancelGenerate | cancelIterate | cancelTimer | eventDestroy | eventForward | eventGenerate | eventIterate | eventTimer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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 `entry` or `exit`.

Input Arguments

`storageID` — Storage index
double

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

`tag` — Tag
character vector

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] = entry(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
```

See Also

cancelDestroy | cancelIterate | cancelTimer | eventDestroy | eventForward |
eventGenerate

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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 `entry`.

Input Arguments

`storageID` — Storage index
double

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

`tag` — Tag
character vector

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] = entry(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
```

See Also

cancelDestroy | cancelForward | cancelGenerate | cancelTimer | eventDestroy | eventForward | eventGenerate | eventIterate | eventTimer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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 `entry`.

Input Arguments

tag — Tag
character vector

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] = entry(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
```

See Also
cancelDestroy | cancelForward | cancelGenerate | cancelIterate | eventDestroy | eventForward | eventGenerate | eventIterate | eventTimer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

destroy

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Event action upon entity destruction

Syntax

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

Description

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

`[events,out1,...]=destroy(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. Entity has these fields:

- `sys` (MATLAB `structure`) — It has these fields:

  - `id` (double) — Entity ID
  - `priority` (`double`) — Entity priority
- `data` — Entity data

`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. Use `matlab.DiscreteEventSystem` class methods to create events. Each event has these fields:

- `type` (`character vector`) — Type of the event
- `delay` (`double`) — Delay before the event
- `priority` (`double`) — Priority of the event
- `Storage` (`double`) — Index of the storage element
- `tag` (`character vector`) — Event tag
- `location` (MATLAB `structure`) — Source or destination location of entity

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 = destroy(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
```

See Also
blocked | entry | exit | generate | getEntityPortsImpl | getEntityStorageImpl | getEntityTypesImpl | iterate | matlab.DiscreteEventSystem | setupEvents | timer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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

Description

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

`entitytype=entityType(name,datatype)` defines a named entity type that takes real values and with specified data type of size 1.

`entitytype=entityType(name,datatype,dimensions)` defines a named entity type that takes real values and with specified data type and size.

`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
character vector

Entity type name.

datatype — Data type (optional)
character vector

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

dimensions — Dimensions (optional)
vector of doubles

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

complexity — Complexity (optional)
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
```

See Also
`getEntityTypesImpl` | `matlab.DiscreteEventSystem`

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

entry

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Event action when entity enters storage element

Syntax

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

Description

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

`[entity,events,out1,...]=entry(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. Entity has these fields:

- `sys` (MATLAB `structure`) — It has these fields:
  - `id` (double) — Entity ID
  - `priority` (double) — Entity priority
- `data` — Entity data

`source` — Source location
MATLAB structure

Source location of entity, such as an input port or a storage element. It has these fields:

- `type` (character vector) — Specify `input` or `storage`
- `index` (double) — Input or storage index

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. See "entity" on page 1-0 .

events — Events
vector of MATLAB structures

Events to be scheduled. Use `matlab.DiscreteEventSystem` class methods to create events. Each event has these fields:

- `type` (`character vector`) — Type of the event
- `delay` (`double`) — Delay before the event
- `priority` (`double`) — Priority of the event
- `Storage` (`double`) — Index of the storage element
- `tag` (`character vector`) — Event tag
- `location` (MATLAB `structure`) — Source or destination location of entity, see "source" on page 1-0

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] = entry(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
```

Create a Custom Entity Storage Block to Delay Entities

This example shows how to use discrete-event System object methods to create a custom entity storage block that has one input port, one output port, and one storage element. The discrete-event

System object is the instantiation of the `matlab.DiscreteEventSystem` class, which allows you to use the implementation and service methods provided by this class. Then, you use the MATLAB Discrete-Event System block to integrate the System object into a SimEvents model.

The custom MATLAB Discrete-Event System block accepts an entity from its input port and forwards it to its output port with a specified delay. For more information, see "Delay Entities with a Custom Entity Storage Block".

```matlab
classdef CustomEntityStorageBlock < matlab.DiscreteEventSystem

    % A custom entity storage block with one input, one output, and one storage.

    % Nontunable properties
    properties (Nontunable)
    % Capacity
        Capacity = 1;
    % Delay
        Delay=4;
    end

    methods (Access=protected)
        function num = getNumInputsImpl(~)
            num = 1;
        end

        function num = getNumOutputsImpl(~)
            num = 1;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityTypes = obj.entityType('Car');
        end

        function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
            inputTypes = {'Car'};
            outputTypes = {'Car'};
        end

        function [storageSpecs, I, O] = getEntityStorageImpl(obj)
            storageSpecs = obj.queueFIFO('Car', obj.Capacity);
            I = 1;
            O = 1;
        end

    end

    methods

        function [entity,event] = CarEntry(obj,storage,entity,source)
            % Specify event actions when entity enters storage.

            event = obj.eventForward('output', 1, obj.Delay);

        end

    end

end
```

See Also

blocked | destroy | exit | generate | getEntityPortsImpl | getEntityStorageImpl | getEntityTypesImpl | iterate | matlab.DiscreteEventSystem | setupEvents | timer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

eventAcquireResource

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Create a resource acquisition event

Syntax

`event = eventAcquireResource(resourceSpec,tag)`

Description

`event = eventAcquireResource(resourceSpec,tag)` creates an event to acquire resources from existing Resource Pool blocks. You can specify names and amount of resources to acquire. For more details, see `resourceSpecification`.

If all the requested resources are not available during the event execution, the acquisition event remains active. When the requested resources become available, the event is rescheduled for immediate execution.

Input Arguments

`resourceSpec` — Specify name and amount of resources for acquisition
array of MATLAB structures

Specify the name and the amount of resources to be acquired by the entities.

`tag` — Identifier tag for the resource acquisition event
character vector

Custom tag of this entity resource acquisition event. You can use the tag to identify an event when multiple events act on the same entity. For more information about managing multiple events, see "Custom Entity Storage Block with Multiple Timer Events".

Output Arguments

`event` — Resource acquisition event
MATLAB structure

Event that acquires resources for the entity.

Examples

Acquire Resources upon Entry

On entity entry to a storage element, an entity acquires one resource of type `Test1`. The tag of this resource acquisition event is `TestTag`.

```
function [entity,events] = entry(obj, storage, entity, source)
% On entity entry, acquire a resource from the specified pool.
```

```
resourceSpec = obj.resourceSpecification('Test1', 1);
event = obj.eventAcquireResource(resourceSpec, 'TestTag');
end
```

Custom Block to Acquire Resources

This example shows how to use resource management methods to create a custom entity storage block in which entities acquire resources from specified Resource Pool blocks.

Suppose that you manage a facility that produces parts from two different materials, material 1 and material 2, to fulfill orders. After a part is produced, it is evaluated for quality assurance.

Two testing methods for quality control are:

- Test 1 is used for parts that are produced from material 1.
- Test 2 is used for parts that are produced from material 2

After the production phase, parts are tagged based on their material to apply the correct test.

For more information, see "Create a Custom Resource Acquirer Block".

```matlab
classdef CustomBlockAcquireResources < matlab.DiscreteEventSystem
    % Custom resource acquire block example.

    methods(Access = protected)

        function num = getNumInputsImpl(obj)
            num = 1;
        end

        function num = getNumOutputsImpl(obj)
            num = 1;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityTypes(1) = obj.entityType('Part');
        end

        function [input, output] = getEntityPortsImpl(obj)
            input  = {'Part'};
            output = {'Part'};
        end

        function [storageSpec, I, O] = getEntityStorageImpl(obj)
            storageSpec(1) = obj.queueFIFO('Part', 1);
            I = 1;
            O = 1;
        end

        function resNames = getResourceNamesImpl(obj)
            % Define the names of the resources to be acquired.
            resNames = obj.resourceType('Part', {'Test1', 'Test2'}) ;
        end

    end

    methods

        function [entity,events] = entry(obj, storage, entity, source)
            % On entity entry, acquire a resource from the specified pool.
            if entity.data.Test == 1
            % If the entity is produced from Material1, request Test1.
            resReq = obj.resourceSpecification('Test1', 1);
            else
            % If the entity is produced from Material2, request Test2.
            resReq = obj.resourceSpecification('Test2', 1);
            end
            % Acquire the resource from the corresponding pool.
            events = obj.eventAcquireResource(resReq, 'TestTag');
        end
```

```
        function [entity,events] = resourceAcquired(obj, storage,...
                              entity, resources, tag)
        % After the resource acquisition, forward the entity to the output.
            events = obj.eventForward('output', storage, 0.0);
        end

    end

end
```

See Also

cancelAcquireResource | eventForward | eventReleaseResource | getResourceNamesImpl
| matlab.DiscreteEventSystem | resourceReleased | resourceSpecification

Topics
"Create a Custom Resource Acquirer Block"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2019a

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 `timer`.

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

See Also
cancelDestroy | cancelForward | cancelGenerate | cancelIterate | cancelTimer | eventForward | eventGenerate | eventIterate | eventTimer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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 entry.

Input Arguments

locationType — Location type
character vector

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] = entry(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);

    % Define events as event1 and event2
    events = [event1, event2];
end
```

Create a Custom Entity Storage Block to Delay Entities

This example shows how to use discrete-event System object methods to create a custom entity storage block that has one input port, one output port, and one storage element. The discrete-event System object is the instantiation of the `matlab.DiscreteEventSystem` class, which allows you to use the implementation and service methods provided by this class. Then, you use the MATLAB Discrete-Event System block to integrate the System object into a SimEvents model.

The custom MATLAB Discrete-Event System block accepts an entity from its input port and forwards it to its output port with a specified delay. For more information, see "Delay Entities with a Custom Entity Storage Block".

```
classdef CustomEntityStorageBlock < matlab.DiscreteEventSystem

    % A custom entity storage block with one input, one output, and one storage.

    % Nontunable properties
    properties (Nontunable)
    % Capacity
        Capacity = 1;
    % Delay
        Delay=4;
    end

    methods (Access=protected)
        function num = getNumInputsImpl(~)
            num = 1;
        end

        function num = getNumOutputsImpl(~)
            num = 1;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityTypes = obj.entityType('Car');
        end

        function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
            inputTypes = {'Car'};
            outputTypes = {'Car'};
        end

        function [storageSpecs, I, O] = getEntityStorageImpl(obj)
            storageSpecs = obj.queueFIFO('Car', obj.Capacity);
            I = 1;
            O = 1;
        end

    end

    methods

        function [entity,event] = CarEntry(obj,storage,entity,source)
            % Specify event actions when entity enters storage.

            event = obj.eventForward('output', 1, obj.Delay);

        end

    end
```

```
end
```

See Also

cancelForward | eventDestroy | eventGenerate | eventIterate | eventTimer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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 `entry`.

Input Arguments

`storageID` — Storage index
double

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

`tag` — Tag
character vector

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.

```matlab
function event = setupEvents(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
```

Create a Custom Block to Generate Entities

This example shows how to create a custom source block that generates entities and to manage discrete states when implementing the discrete-event System object methods.

For more information, see "Custom Entity Generator Block with Signal Input and Signal Output".

```matlab
classdef CustomEntityStorageBlockGeneration < matlab.DiscreteEventSystem...
                                    & matlab.system.mixin.Propagates
    % A custom entity generator block.

    % Nontunable properties
    properties (Nontunable)
        % Generation period
        period = 1;
    end

    properties(DiscreteState)
        % Entity priority
        priority;
        % Entity value
        value;
    end

    % Discrete-event algorithms
    methods
        function [events, out1] = setupEvents(obj)
            % Set up entity generation events at simulation start.
            events = obj.eventGenerate(1,'mygen',obj.period,obj.priority);
            % Set up the initial value of the output signal.
            out1 = 10;
        end

        function [entity,events,out1] = generate(obj,storage,entity,tag,in1)
            % Specify event actions when entity is generated in storage.
            entity.data = obj.value;
            % The priority value is assigned from the input signal.
            obj.priority = in1;
            % Output signal is the assigned priority value.
            out1 = obj.priority;
            events = [obj.eventForward('output',1,0) ...
                        obj.eventGenerate(1,'mygen',obj.period,obj.priority)];
        end
    end

    methods(Access = protected)

        function entityTypes = getEntityTypesImpl(obj)
            entityTypes = obj.entityType('Material');
        end

        function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
            % Specify entity input and output ports. Return entity types at
            % a port as strings in a cell array. Use empty string to
            % indicate a data port.
            inputTypes = {''};
            outputTypes = {'Material',''};
```

```matlab
        end

        function resetImpl(obj)
            % Initialize / reset discrete-state properties.
            obj.priority = 10;
            obj.value = 1:12;
        end

        function [storageSpecs, I, O] = getEntityStorageImpl(obj)
            storageSpecs = obj.queueFIFO('Material', 1);
            I = 0;
            O = [1 0];
        end

        function num = getNumInputsImpl(obj)
            % Define total number of inputs for system with optional
            % inputs.
            num = 1;
        end

        function num = getNumOutputsImpl(~)
            % Define total number of outputs.
            num = 2;
        end
        function [out1 out2] = getOutputSizeImpl(obj)
            % Return size for each output port.
            out1 = [1 12];
            out2 = 1;
        end

        function [out1 out2] = getOutputDataTypeImpl(obj)
            % Return data type for each output port.
            out1 = "double";
            out2 = "double";
        end

        function [out1 out2] = isOutputComplexImpl(obj)
            % Return true for each output port with complex data.
            out1 = false;
            out2 = false;
        end

        function [sz,dt,cp] = getDiscreteStateSpecificationImpl(obj,name)
            % Return size, data type, and complexity of discrete-state
            % specified in name.
            switch name
                case 'priority'
                    sz = [1 1];
                case 'value'
                    sz = [1 12];
            end
            dt = "double";
            cp = false;
        end
    end
end
```

See Also

cancelGenerate | cancelTimer | eventDestroy | eventForward | eventIterate | eventTimer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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 `exit`.

Input Arguments

`storageID` — Storage index
double

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

`tag` — Tag
character vector

Custom tag of this entity iterate event.

`priority` — Priority (optional)
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 = exit(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
```

Custom Entity Storage Block with Iterate Event

In this example, a custom block allows entities to enter its storage element through its input port. The storage element is a priority queue that sorts the entities based on their `Diameter` attribute in ascending order. Every entity entry to the block's storage invokes an iteration event to display the diameter and the position of each entity in the storage.

For more information, see "Create a Custom Entity Storage Block with Iteration Event".

```matlab
classdef CustomEntityStorageBlockIteration < matlab.DiscreteEventSystem

    % A custom entity storage block with one input port and one storage element.

    % Nontunable properties
    properties (Nontunable)
        % Capacity
        Capacity = 5;
    end
    % Create the storage element with one input and one storage.
    methods (Access=protected)

        function num = getNumInputsImpl(obj)
            num = 1;
        end

        function num = getNumOutputsImpl(obj)
            num = 0;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityType1 = obj.entityType('Wheel');
            entityTypes = entityType1;
        end

        function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
            inputTypes = {'Wheel'};
            outputTypes={};

        end

        function [storageSpecs, I, O] = getEntityStorageImpl(obj)
            storageSpecs = obj.queuePriority('Wheel',obj.Capacity, 'Diameter','ascending');
            I = 1;
            O = [];

        end

    end
    % Entity entry event action
    methods

        function [entity, event] = WheelEntry(obj,storage,entity, source)
            % Entity entry invokes an iterate event.
            event = obj.eventIterate(1, '');
        end

        % The itarate event action
        function [entity,event,next] = WheelIterate(obj,storage,entity,tag,cur)
            % Display wheel id, position in the storage, and diameter.
            coder.extrinsic('fprintf');
            fprintf('Wheel id %d, Current position %d, Diameter %d\n', ...
                entity.sys.id, cur.position, entity.data.Diameter);
            if cur.size == cur.position
                fprintf('End of Iteration \n')
            end
            next = true;
            event=[];
```

```
        end
    end
end
```

See Also
cancelIterate | cancelTimer | eventDestroy | eventForward | eventGenerate |
eventTimer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

eventReleaseAllResources

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Create an event to release all resources acquired by an entity

Syntax

```
event = eventReleaseAllResources(tag)
```

Description

`event = eventReleaseAllResources(tag)` creates an event to release all the resources acquired by an entity.

Input Arguments

`tag` — Identifier tag for the resource release event
character vector

Custom tag of this entity resource release event. You can use the tag to identify an event when multiple events act on the same entity. For more information about managing multiple events, see "Custom Entity Storage Block with Multiple Timer Events".

Output Arguments

`event` — Resource release event
MATLAB structure

Event that releases all resources from the entity.

Examples

On entity entry to a storage element, an entity releases all of the previously acquired resources. The tag of this resource acquisition event is `ReleaseAll`.

```
function [entity, event] = entry(obj, storage, entity, source)
     event = obj.eventReleaseAllResources('releaseAll');
end
```

See Also
`eventForward` | `eventReleaseResource` | `getResourceNamesImpl` | `matlab.DiscreteEventSystem` | `resourceReleased`

Topics
"Create a Custom Resource Acquirer Block"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2019a

eventReleaseResource

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Create an event to release previously acquired resources

Syntax

```
event = eventReleaseResource(resourceSpec,tag)
```

Description

`event = eventReleaseResource(resourceSpec,tag)` creates an event for entities to release previously acquired resources. You can specify the type and amount of resources to release. For more details, see `resourceSpecification`.

If the amount of resources to be released is larger than the previously acquired resources, then all the resources are released.

Input Arguments

`resourceSpec` — Specify name and amount of resources for release
array of MATLAB structures

Specify the name and the amount of resources to be released by the entities.

`tag` — Identifier tag for the resource release event
character vector

Custom tag of this entity resource release event. You can use the tag to identify an event when multiple events act on the same entity. For more information about managing multiple events, see "Custom Entity Storage Block with Multiple Timer Events".

Output Arguments

`event` — Resource release event
MATLAB structure

Event that releases resources from the entity.

Examples

Acquire Resources on Entry

On entity entry to a storage element, an entity releases one resource of type `Test1`. The tag of this resource acquisition event is `myTag`.

```
function [entity,events] = entry(obj, storage, entity, source)
% On entity entry, release a resource from the specified pool.
```

```
resourceSpec = obj.resourceSpecification('Test1', 1);
event = obj.eventReleaseResource(resourceSpec, 'myTag');
end
```

See Also

cancelAcquireResource | eventAcquireResource | eventForward | getResourceNamesImpl | matlab.DiscreteEventSystem | resourceAcquired | resourceSpecification

Topics
"Create a Custom Resource Acquirer Block"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2019a

eventTestEntry

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Create an event to indicate that the acceptance policy for the storage has changed and the storage retests arriving entities

Syntax

```
event = eventTestEntry(storageId)
```

Description

`event = eventTestEntry(storageId)` creates an event to retest entities arriving at storage.

Input Arguments

storageId — Index of storage element
scalar

Index of the storage element that is to be unblocked, specified as a `scalar`.

Data Types: `double`

Output Arguments

event — Unblock
MATLAB structure

Event that unblocks a storage or input port for processing entities, specified as a MATLAB structure.

Examples

Event Action to Retest Entity Entry

Retest entity entry to storage.

```
function [entity,events] = exit(obj,storage,entity,dst)
    % Indicates that more entities can be accepted and acceptance policy has changed
    events = [events,obj.eventTestEntry(storage)];
end
```

See Also
`iterate` | `matlab.DiscreteEventSystem`

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2018a

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 `entry`.

Input Arguments

tag — Tag
character vector

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] = entry(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
```

Custom Block with Timer Events

This example uses a custom entity storage block with one input, two outputs, and a storage element. An entity of type `Part` with `TimeOut` attribute enters the storage of the custom block to be

processed. `TimeOut` determines the maximum allowed processing time of the parts. When a part enters the storage, two timer events are activated. One timer tracks the processing time of the part in the oven. When this timer expires, the entity is forwarded to output 1. Another timer acts as a fail-safe and tracks if the maximum allowed processing time is exceeded or not. When this timer expires, the process is terminated and the entity is forwarded to the output 2.

For more information, see "Custom Entity Storage Block with Multiple Timer Events".

```matlab
classdef CustomEntityStorageBlockTimer < matlab.DiscreteEventSystem

    % A custom entity storage block with one input port, two output ports, and one storage.

    % Nontunable properties
    properties (Nontunable)
    % Capacity
        Capacity = 1;
    end

    methods (Access=protected)

        function num = getNumInputsImpl(~)
            num = 1;
        end

        function num = getNumOutputsImpl(~)
            num = 2;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityTypes = obj.entityType('Part');
        end

        function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
            inputTypes = {'Part'};
            outputTypes = {'Part' 'Part'};
        end

        function [storageSpecs, I, O] = getEntityStorageImpl(obj)
            storageSpecs = obj.queueFIFO('Part', obj.Capacity);
            I = 1;
            O = [1 1];
        end

    end


    methods

        function [entity,event] = PartEntry(obj,storage,entity,source)
            % Specify event actions when entity enters storage.
            ProcessingTime=randi([1 15]);
            event1 = obj.eventTimer('TimeOut', entity.data.TimeOut);
            event2 = obj.eventTimer('ProcessComplete', ProcessingTime);
            event = [event1 event2];
        end

        function [entity, event] = timer(obj,storage,entity,tag)
            % Specify event actions for when scheduled timer completes.
            event = obj.initEventArray;
            switch tag
                case 'ProcessComplete'
                    event = obj.eventForward('output', 1, 0);
                case 'TimeOut'
                    event = obj.eventForward('output', 2, 0);
            end

        end

    end

end
```

See Also

cancelDestroy | cancelForward | cancelGenerate | cancelIterate | cancelTimer | eventDestroy | eventForward | eventGenerate | eventIterate

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

exit

Class: matlab.DiscreteEventSystem
Package: matlab

Event action before entity exit from storage

Syntax

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

Description

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

`[events,out1,...]=exit(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. Entity has these fields:

- `sys` (MATLAB `structure`) — It has these fields:
    - `id` (double) — Entity ID
    - `priority` (double) — Entity priority
- `data` — Entity data

`destination` — Destination
MATLAB structure

Destination of entity, such as an output port or a storage element. It has these fields:

- `type` (character vector) — Specify `output`, `storage`, or `extract`
- `index` (double) — Output or storage index

The `type` is specified as `extract` if an entity is being extracted from a Discrete-Event System block.

`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. Use `matlab.DiscreteEventSystem` class methods to create events. Each event has these fields:

- `type` (character vector) — Type of the event
- `delay` (double) — Delay before the event
- `priority` (double) — Priority of the event
- `Storage` (double) — Index of the storage element
- `tag` (character vector) — Event tag
- `location` (MATLAB `structure`) — Source or destination location of entity

`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 = exit(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
```

See Also

blocked | destroy | entry | generate | getEntityPortsImpl | getEntityStorageImpl | getEntityTypesImpl | iterate | matlab.DiscreteEventSystem | setupEvents | timer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

generate

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Event action upon entity creation

Syntax

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

Description

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

`[entity,events,out1,...]=generate(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. Entity has these fields:

- `sys` (MATLAB `structure`) — It has these fields:
  - `id` (double) — Entity ID
  - `priority` (double) — Entity priority
- `data` — Entity data

`tag` — Tag
character vector

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. Use `matlab.DiscreteEventSystem` class methods to create events. Each event has these fields:

- `type` (`character vector`) — Type of the event
- `delay` (`double`) — Delay before the event
- `priority` (`double`) — Priority of the event
- `Storage` (`double`) — Index of the storage element
- `tag` (`character vector`) — Event tag
- `location` (MATLAB `structure`) — Source or destination location of entity

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] = generate(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
```

Create a Custom Block to Generate Entities

This example shows how to create a custom source block that generates entities and to manage discrete states when implementing the discrete-event System object methods.

For more information, see "Custom Entity Generator Block with Signal Input and Signal Output".

```
classdef CustomEntityStorageBlockGeneration < matlab.DiscreteEventSystem...
                                  & matlab.system.mixin.Propagates
    % A custom entity generator block.

    % Nontunable properties
    properties (Nontunable)
        % Generation period
        period = 1;
    end

    properties(DiscreteState)
        % Entity priority
        priority;
        % Entity value
        value;
    end

    % Discrete-event algorithms
    methods
        function [events, out1] = setupEvents(obj)
            % Set up entity generation events at simulation start.
            events = obj.eventGenerate(1,'mygen',obj.period,obj.priority);
            % Set up the initial value of the output signal.
            out1 = 10;
        end

        function [entity,events,out1] = generate(obj,storage,entity,tag,in1)
            % Specify event actions when entity is generated in storage.
            entity.data = obj.value;
            % The priority value is assigned from the input signal.
            obj.priority = in1;
            % Output signal is the assigned priority value.
            out1 = obj.priority;
            events = [obj.eventForward('output',1,0) ...
                        obj.eventGenerate(1,'mygen',obj.period,obj.priority)];
        end
    end

    methods(Access = protected)

        function entityTypes = getEntityTypesImpl(obj)
            entityTypes = obj.entityType('Material');
        end

        function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
            % Specify entity input and output ports. Return entity types at
            % a port as strings in a cell array. Use empty string to
            % indicate a data port.
            inputTypes = {''};
            outputTypes = {'Material',''};
        end

        function resetImpl(obj)
            % Initialize / reset discrete-state properties.
            obj.priority = 10;
            obj.value = 1:12;
        end

        function [storageSpecs, I, O] = getEntityStorageImpl(obj)
            storageSpecs = obj.queueFIFO('Material', 1);
            I = 0;
            O = [1 0];
        end

        function num = getNumInputsImpl(obj)
            % Define total number of inputs for system with optional
            % inputs.
            num = 1;
```

```
        end

        function num = getNumOutputsImpl(~)
            % Define total number of outputs.
            num = 2;
        end
        function [out1 out2] = getOutputSizeImpl(obj)
            % Return size for each output port.
            out1 = [1 12];
            out2 = 1;
        end

        function [out1 out2] = getOutputDataTypeImpl(obj)
            % Return data type for each output port.
            out1 = "double";
            out2 = "double";
        end

        function [out1 out2] = isOutputComplexImpl(obj)
            % Return true for each output port with complex data.
            out1 = false;
            out2 = false;
        end

        function [sz,dt,cp] = getDiscreteStateSpecificationImpl(obj,name)
            % Return size, data type, and complexity of discrete-state
            % specified in name.
            switch name
                case 'priority'
                    sz = [1 1];
                case 'value'
                    sz = [1 12];
            end
            dt = "double";
            cp = false;
        end
    end
end
```

See Also

blocked | destroy | entry | exit | getEntityPortsImpl | getEntityStorageImpl | getEntityTypesImpl | iterate | matlab.DiscreteEventSystem | setupEvents | timer

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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 character vectors

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

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

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

`outputTypes` — Output types
cell vector of character vectors

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 that specifies type of the *N*th output port.

- If the port is an entity port, the character vector indicates the entity type name of this port. The name must match one of the entity types specified in `getEntityTypesImpl`.
- If the port is a signal port, the character vector 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
```

See Also
getEntityStorageImpl | getEntityTypesImpl | matlab.DiscreteEventSystem

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

getEntityStorageImpl

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Define entity storage elements of discrete-event system

Syntax

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

Description

`[storageSpecs,I,O]=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 Nth 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
cell array

Define connections between input ports and entity storage elements as a cell array. The length of the cell array must match the number of input ports of this discrete-event system. The Nth element of the cell array defines the connection between the Nth 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.

You can connect multiple entity input ports to a common storage element.

`O` — Connections between output ports and entity storage elements
cell array

Define connections between output ports and entity storage elements as a cell array. The length of the cell array must match the number of output ports of this discrete-event system. The Mth element of the cell array defines the connections between the Mth output port and any entity storage elements. If the output port is an entity port, specify one of these:

- A scalar indicating a single connection from a storage element to the output port.

- A vector indicating multiple connections from multiple storage elements to the output port.

If the output port is a signal port, the element takes a value of zero.

You can connect multiple entity output ports to a common storage element.

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
```

See Also
getEntityTypesImpl | matlab.DiscreteEventSystem | queueFIFO | queueLIFO | queuePriority | queueSysPriority

Topics
"Delay Entities with a Custom Entity Storage Block"
"Create a Custom Entity Storage Block with Iteration Event"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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
```

See Also

entityType | getEntityPortsImpl | getEntityStorageImpl |
matlab.DiscreteEventSystem

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

getResourceNamesImpl

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Define resource pools from which to acquire resources

Syntax

`resourceNames = getResourceNamesImpl(obj)`

Description

`resourceNames = getResourceNamesImpl(obj)` defines resource pools from which the discrete event system acquires resources.

Input Arguments

obj — Discrete-event System object
MATLAB object

Discrete-event System object.

Output Arguments

resourceNames — Resource names
vector of MATLAB structures

Resource pools from which to acquire resources, specified as a vector of MATLAB structures. Use `resourceType` method to create this array.

Examples

Define Resource Pools

Use this method together with `resourceType` to specify the resources of types `Test1` and `Test2` to be acquired by the entity type `Part`.

```
function resourceNames = getResourceNamesImpl(obj)
    % Define the names of the resources to be acquired.
    resourceNames = obj.resourceType('Part', {'Test1', 'Test2'}) ;
end
```

See Also
`cancelAcquireResource` | `eventForward` | `eventReleaseResource` | `matlab.DiscreteEventSystem` | `resourceAcquired` | `resourceType`

Topics
"Create a Custom Resource Acquirer Block"

"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2019a

initEventArray

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Initialize event array

Syntax

`event = initEventArray()`

Description

`event = initEventArray()` creates an empty array of event structures, to initialize the return of an event action method such as `matlab.DiscreteEventSystem.entry`. This method enables you to append elements to the array in the MATLAB Discrete-Event System block when the `Code generation` is selected for the **Simulate using** parameter.

Output Arguments

event — Array of event structures
Array

Array of event structures, specified as a MATLAB structures.

Examples

Initialize Returned Event Array

Initialize returned event array for the `exit` method.

```
function events = exit(obj, ~, entity, ~)
        events = obj.initEventArray;
        if entity.data == 1
            events = obj.eventTimer('exit', 0);
        end
end
```

See Also
`getEntityStorageImpl` | `queueLIFO` | `queuePriority` | `queueSysPriority`

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2017b

initResourceArray

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Initialize a resource specification array

Syntax

```
resArray = initResourceArray()
```

Description

`resArray = initResourceArray()` initializes an empty array of `resourceSpecification`. This method enables you to append elements to the array in the MATLAB Discrete-Event System block when you select the `Code generation` for the **Simulate using** parameter.

Output Arguments

resArray — Resource specifications
`Array`

Resource specifications specified as an empty array.

Examples

Initialize Resource Array

An entity entry to the storage element invokes two timer events. `resourceSpecification` defines the type and the amount of resources an entity acquires. The entity acquires a resource of type `Test1` if the timer with tag `ProcessComplete` expires. The entity acquires a resource of type `Test2` if the timer with tag `TimeOut` expires. The `resRequest` array is initialized by the `initResourceArray` method for code generation.

```
function [entity,event] = entry(obj,storage,entity,source)
  % Specify event actions when entity enters storage.
  ProcessingTime=randi([1 15]);
  % Define two timer events.
  event1 = obj.eventTimer('TimeOut',10);
  event2 = obj.eventTimer('ProcessComplete', ProcessingTime);
  event=[event1 event2];
end

function [entity, event] = timer(obj,storage,entity,tag)
  % Specify event actions when a timer expires.
  resRequest = obj.initResourceArray();
  switch tag
    case 'ProcessComplete'
      resRequest = obj.resourceSpecification('Test1', 1);
    case 'TimeOut'
      resRequest = obj.resourceSpecification('Test2', 1);
  end
```

```
  event = obj.eventAcquireResource(resRequest, 'MyResourceAcquireEvent');
end
```

See Also

eventForward | eventReleaseResource | getResourceNamesImpl |
matlab.DiscreteEventSystem | resourceReleased | resourceSpecification

Topics
"Create a Custom Resource Acquirer Block"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2019a

iterate

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Event action when entity iterates

Syntax

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

Description

`[entity,events,next]=iterate(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,...]=iterate(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. Entity has these fields:

- `sys` (MATLAB `structure`) — It has these fields:
  - `id` (double) — Entity ID
  - `priority` (double) — Entity priority
- `data` — Entity data

`tag` — Tag
character vector

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. Use `matlab.DiscreteEventSystem` class methods to create events. Each event has these fields:

- `type` (`character vector`) — Type of the event
- `delay` (`double`) — Delay before the event
- `priority` (`double`) — Priority of the event
- `Storage` (`double`) — Index of the storage element
- `tag` (`character vector`) — Event tag
- `location` (MATLAB `structure`) — Source or destination location of entity

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.

```matlab
function [entity,events,next] = iterate(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
```

Custom Entity Storage Block with Iteration Event

In this example, a custom block allows entities to enter its storage element through its input port. The storage element is a priority queue that sorts the entities based on their `Diameter` attribute in ascending order. Every entity entry to the block's storage invokes an iteration event to display the diameter and the position of each entity in the storage.

For more information, see "Create a Custom Entity Storage Block with Iteration Event".

```matlab
classdef CustomEntityStorageBlockIteration < matlab.DiscreteEventSystem

    % A custom entity storage block with one input port and one storage element.

    % Nontunable properties
    properties (Nontunable)
        % Capacity
        Capacity = 5;
    end
    % Create the storage element with one input and one storage.
    methods (Access=protected)

        function num = getNumInputsImpl(obj)
            num = 1;
        end

        function num = getNumOutputsImpl(obj)
            num = 0;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityType1 = obj.entityType('Wheel');
            entityTypes = entityType1;
        end

        function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
            inputTypes = {'Wheel'};
            outputTypes={};

        end

        function [storageSpecs, I, O] = getEntityStorageImpl(obj)
            storageSpecs = obj.queuePriority('Wheel',obj.Capacity, 'Diameter','ascending');
            I = 1;
            O = [];

        end

    end
```

```
% Entity entry event action
methods

    function [entity, event] = WheelEntry(obj,storage,entity, source)
        % Entity entry invokes an iterate event.
        event = obj.eventIterate(1, '');
    end

    % The itarate event action
    function [entity,event,next] = WheelIterate(obj,storage,entity,tag,cur)
        % Display wheel id, position in the storage, and diameter.
        coder.extrinsic('fprintf');
        fprintf('Wheel id %d, Current position %d, Diameter %d\n', ...
            entity.sys.id, cur.position, entity.data.Diameter);
        if cur.size == cur.position
            fprintf('End of Iteration \n')
        end
        next = true;
        event=[];
    end

end

end
```

See Also

blocked | destroy | entry | exit | generate | getEntityPortsImpl | getEntityStorageImpl | getEntityTypesImpl | matlab.DiscreteEventSystem | setupEvents | timer

Topics

"Create a Custom Entity Storage Block with Iteration Event"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

modified

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Event action upon entity modification by the Entity Find block

Syntax

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

Description

`[events] = modified(obj,storage,entity)` specifies event actions of the object after an entity is modified.

`[events, out1, ...] = modified(obj,storage,entity,in1,...)` specifies 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 where the entity is being modified.

`entity` — Entity
MATLAB structure

Entity that is being modified. Entity has these fields:

- `sys` (MATLAB `structure`) — It has these fields:

  - `id` (double) — Entity ID
  - `priority` (double) — Entity priority
- `data` — Entity data

`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 after the method returns. Use `matlab.DiscreteEventSystem` class methods to create events. Each event has these fields:

- `type` (`character vector`) — Type of the event
- `delay` (`double`) — Delay before the event
- `priority` (`double`) — Priority of the event
- `Storage` (`double`) — Index of the storage element
- `tag` (`character vector`) — Event tag
- `location` (MATLAB `structure`) — Source or destination location of entity

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 Modification

Specify event action to be performed after entity modification in a storage

```
function events = modified(obj,storage,entity)
    events = [];
    % If the delay attribute of the entity exceeds 100, destroy the entity
    if entity.data.delay > 100
    events = obj.destroy();
    end
end
```

See Also
`blocked` | `destroy` | `entry` | `generate` | `iterate` | `matlab.DiscreteEventSystem` | `setupEvents` | `timer`

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2018b

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 `getEntityStorageImpl` method.

Note This page describes the `queueFIFO` method to create a custom discrete-event system block. See Entity Queue block provided by the SimEvents library to use a queue with FIFO sorting policy.

Input Arguments

`entityType` — Entity type
character vector

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);
```

Create a Custom Entity Storage Block to Delay Entities

This example shows how to use discrete-event System object methods to create a custom entity storage block that has one input port, one output port, and one storage element. The discrete-event System object is the instantiation of the `matlab.DiscreteEventSystem` class, which allows you to use the implementation and service methods provided by this class. Then, you use the MATLAB Discrete-Event System block to integrate the System object into a SimEvents model. The custom MATLAB Discrete-Event System block accepts an entity from its input port and forwards it to its output port with a specified delay. For more information, see "Delay Entities with a Custom Entity Storage Block".

```
classdef CustomEntityStorageBlock < matlab.DiscreteEventSystem

    % A custom entity storage block with one input, one output, and one storage.

    % Nontunable properties
    properties (Nontunable)
    % Capacity
        Capacity = 1;
    % Delay
        Delay=4;
    end

    methods (Access=protected)
        function num = getNumInputsImpl(~)
            num = 1;
        end

        function num = getNumOutputsImpl(~)
            num = 1;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityTypes = obj.entityType('Car');
        end

        function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
            inputTypes = {'Car'};
            outputTypes = {'Car'};
        end

        function [storageSpecs, I, O] = getEntityStorageImpl(obj)
            storageSpecs = obj.queueFIFO('Car', obj.Capacity);
            I = 1;
            O = 1;
        end

    end

    methods

        function [entity,event] = CarEntry(obj,storage,entity,source)
            % Specify event actions when entity enters storage.

             event = obj.eventForward('output', 1, obj.Delay);

        end

    end

end
```

See Also

getEntityStorageImpl | queueLIFO | queuePriority | queueSysPriority

Topics
"Delay Entities with a Custom Entity Storage Block"
"Create a Custom Entity Storage Block with Iteration Event"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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 `getEntityStorageImpl` method.

Input Arguments

`entityType` — Entity type
character vector

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);
```

See Also

`getEntityStorageImpl` | `queueFIFO` | `queuePriority` | `queueSysPriority`

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

queuePriority

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Define priority queue storage

Syntax

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

Description

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

Input Arguments

`entityType` — Entity type
character vector

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
character vector

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

`order` — Sorting order
character vector

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');
```

Custom Entity Storage Block with Iteration Event

In this example, a custom block allows entities to enter its storage element through its input port. The storage element is a priority queue that sorts the entities based on their `Diameter` attribute in ascending order. Every entity entry to the block's storage invokes an iteration event to display the diameter and the position of each entity in the storage.

For more information, see "Create a Custom Entity Storage Block with Iteration Event".

```
classdef CustomEntityStorageBlockIteration < matlab.DiscreteEventSystem

    % A custom entity storage block with one input port and one storage element.

    % Nontunable properties
    properties (Nontunable)
        % Capacity
        Capacity = 5;
    end
    % Create the storage element with one input and one storage.
    methods (Access=protected)

        function num = getNumInputsImpl(obj)
            num = 1;
        end

        function num = getNumOutputsImpl(obj)
            num = 0;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityType1 = obj.entityType('Wheel');
            entityTypes = entityType1;
        end

        function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
            inputTypes = {'Wheel'};
            outputTypes={};

        end

        function [storageSpecs, I, O] = getEntityStorageImpl(obj)
            storageSpecs = obj.queuePriority('Wheel',obj.Capacity, 'Diameter','ascending');
            I = 1;
            O = [];

        end

    end
    % Entity entry event action
    methods

        function [entity, event] = WheelEntry(obj,storage,entity, source)
            % Entity entry invokes an iterate event.
            event = obj.eventIterate(1, '');
        end
```

```matlab
        % The itarate event action
        function [entity,event,next] = WheelIterate(obj,storage,entity,tag,cur)
            % Display wheel id, position in the storage, and diameter.
            coder.extrinsic('fprintf');
            fprintf('Wheel id %d, Current position %d, Diameter %d\n', ...
                entity.sys.id, cur.position, entity.data.Diameter);
            if cur.size == cur.position
                fprintf('End of Iteration \n')
            end
            next = true;
            event=[];
        end

    end

end
```

See Also

getEntityStorageImpl | queueFIFO | queueLIFO | queueSysPriority

Topics

"Create a Custom Entity Storage Block with Iteration Event"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

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 `getEntityStorageImpl` method.

Input Arguments

`entityType` — Entity type
character vector

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
character vector

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');
```

See Also

getEntityStorageImpl | queueFIFO | queueLIFO | queuePriority

Topics

"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

resourceAcquired

Class: matlab.DiscreteEventSystem
Package: matlab

Event action upon successful resource acquisition

Syntax

```
[entity,event,out1,...] = resourceAcquired(obj,storage,entity,resources,tag,
in1,...)
```

Description

`[entity,event,out1,...] = resourceAcquired(obj,storage,entity,resources,tag,
in1,...)` specifies event action for a discrete-event System object upon successful acquisition of a resource. Resource acquisition is successful only if all of the specified resources are acquired.

Input Arguments

`obj` — Discrete-event System object
MATLAB object

Discrete-event System object.

`storage` — Storage that the entity resides in
double

Index of the storage element.

`entity` — Entity that is acquiring the resources
MATLAB structure

Entity that acquires the resource. Entity has these fields:

- `sys` (MATLAB `structure`) consisting of:

  - `id` (double) — Entity ID
  - `priority` (double) — Entity priority
- `data` — Entity data

`resources` — Acquired resources
MATLAB structure

An array of structures that specifies the resources that have been acquired.

`tag` — Tag of the resource acquisition event
character vector

Tag of the currently executing resource acquisition event.

in1 — First data input
character vector

First data input.

Output Arguments

entity — Entity with changed value
MATLAB structure

Entity acquiring the resource.

event — Events to be scheduled
vector of MATLAB structures

Events to be scheduled. Use `matlab.DiscreteEventSystem` class methods to create events. Each event has these fields:

- `type` (`character vector`) — Type of the event
- `delay` (`double`) — Delay before the event
- `priority` (`double`) — Priority of the event
- `storage` (`double`) — Index of the storage element
- `tag` (`character vector`) — Event tag
- `location` (MATLAB `structure`) — Source or destination (see "source" on page 1-0   )

out1 — First data output
character vector

First data output.

Examples

Event Action on Resource Acquisition

Suppose that an entity acquires resources successfully with a scheduled `eventAcquireResource` and the tag of this event is `MyResourceAcquireEvent`. Then this acquisition invokes the `resourceAcquired` method to forward entities to the output.

```
function [entity,events] = entry(obj, storage, entity, source)
  % On entry, acquire one resource of type Resource1.
  resRequest = obj.resourceSpecification('Resource1', 1);
  events = obj.eventAcquireResource(resRequest, 'MyResourceAcquireEvent');
end

function [entity,events] = resourceAcquired(obj, storage,...
                           entity, resources, tag)
  % After resource acquisition, forward the entity to the output.
  events = obj.eventForward('output', storage, 0.0);
end
```

Custom Resource Acquirer

This example shows how to use resource management methods to create a custom entity storage block in which entities acquire resources from specified Resource Pool blocks.

Suppose that you manage a facility that produces parts from two different materials, material 1 and material 2, to fulfill orders. After a part is produced, it is evaluated for quality assurance.

Two testing methods for quality control are:

- Test 1 is used for parts that are produced from material 1.
- Test 2 is used for parts that are produced from material 2

After the production phase, parts are tagged based on their material to apply the correct test.

For more information, see "Create a Custom Resource Acquirer Block".

```matlab
classdef CustomBlockAcquireResources < matlab.DiscreteEventSystem
    % Custom resource acquire block example.

    methods(Access = protected)

        function num = getNumInputsImpl(obj)
            num = 1;
        end

        function num = getNumOutputsImpl(obj)
            num = 1;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityTypes(1) = obj.entityType('Part');
        end

        function [input, output] = getEntityPortsImpl(obj)
            input  = {'Part'};
            output = {'Part'};
        end

        function [storageSpec, I, O] = getEntityStorageImpl(obj)
            storageSpec(1) = obj.queueFIFO('Part', 1);
            I = 1;
            O = 1;
        end

        function resNames = getResourceNamesImpl(obj)
            % Define the names of the resources to be acquired.
            resNames = obj.resourceType('Part', {'Test1', 'Test2'}) ;
        end

    end

    methods

        function [entity,events] = entry(obj, storage, entity, source)
            % On entity entry, acquire a resource from the specified pool.
            if entity.data.Test == 1
            % If the entity is produced from Material1, request Test1.
            resReq = obj.resourceSpecification('Test1', 1);
            else
            % If the entity is produced from Material2, request Test2.
            resReq = obj.resourceSpecification('Test2', 1);
            end
            % Acquire the resource from the corresponding pool.
            events = obj.eventAcquireResource(resReq, 'TestTag');
        end

        function [entity,events] = resourceAcquired(obj, storage,...
                                    entity, resources, tag)
        % After the resource acquisition, forward the entity to the output.
            events = obj.eventForward('output', storage, 0.0);
        end

    end

end
```

See Also

cancelAcquireResource | eventAcquireResource | eventForward | getResourceNamesImpl | matlab.DiscreteEventSystem | resourceReleased

Topics
"Create a Custom Resource Acquirer Block"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2019a

resourceReleased

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Event action upon successful resource release

Syntax

```
[entity,event,out1,...] = resourceReleased(obj,storage,entity,resources,tag,
in1,...)
```

Description

`[entity,event,out1,...] = resourceReleased(obj,storage,entity,resources,tag,
in1,...)` specifies event actions for a discrete-event System object upon successful resource
release.

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 releasing the resource. Entity has these fields:

- `sys` (MATLAB `structure`) consisting of:
  - `id` (double) — Entity ID
  - `priority` (double) — Entity priority
- `data` — Entity data

`resources` — Released resources
MATLAB structure

An array of structures that specifies the resources that have been released.

`tag` — Tag of the resource release event
character vector

Tag of the currently executing resource release event.

in1 — First data input
character vector

First data input.

Output Arguments

entity — Entity
MATLAB structure

Entity releasing the resource.

event — Event
vector of MATLAB structures

Events to be scheduled. Use `matlab.DiscreteEventSystem` class methods to create events. Each event has these fields:

- `type` (`character vector`) — Type of the event
- `delay` (`double`) — Delay before the event
- `priority` (`double`) — Priority of the event
- `storage` (`double`) — Index of the storage element
- `tag` (`character vector`) — Event tag
- `location` (MATLAB `structure`) — Source or destination location of entity (see "source" on page 1-0   )

out1 — First data output
character vector

First data output.

Examples

Event Action on Resource Release

Suppose that an entity releases resources successfully with a scheduled `eventReleaseResource` method and the tag of this event is `MyResourceAcquireEvent`. The successful release of the resources invokes the `resourceReleased` method to forward the entity to the output.

```
function [entity,events] = entry(obj, storage, entity, source)
  % On entry, release one resource of type Resource1.
  resRequest = obj.resourceSpecification('Resource1', 1);
  events = obj.eventReleaseResource(resRequest, 'MyResourceAcquireEvent');
end

function [entity,events] = resourceReleased(obj, storage,...
                          entity, resources, tag)
  % After resource release, forward the entity to the output.
  events = obj.eventForward('output', storage, 0.0);
end
```

See Also
cancelAcquireResource | eventForward | eventReleaseResource | getResourceNamesImpl
| matlab.DiscreteEventSystem | resourceAcquired

Topics
"Create a Custom Resource Acquirer Block"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2019a

resourceSpecification

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Create specifications for a resource acquisition or a resource release event

Syntax

```
resRequest = resourceSpecification(resource,amount)
```

Description

`resRequest = resourceSpecification(resource,amount)` specifies the names and the amount of the resources for the `eventAcquireResource` or the `eventReleaseResource` requests.

For example, this code specifies one unit of `Resource1` and two units of `Resource2`.

```
resRequest = obj.resourceSpecification('Resource1', 1);
resRequest = [resReq obj.resourceSpecification('Resource2', 2)];
```

If you specify an amount for the `eventReleaseResource` larger than the amount that was acquired earlier, all the previously acquired resources are released.

For example, suppose that an entity previously acquired three resources of type `Resource1` and four resources of type `Resource2`. This code specifies the amounts for `eventReleaseResource`.

```
resRequest = obj.resourceSpecification('Resource1', 2);
resRequest = [resReq obj.resourceSpecification('Resource2', 5)];
    event = eventReleaseResource(resReq, 'relinquish');
```

After the release, the entity has one resource of type `Resource1` and zero resources of type `Resource2`.

You can specify the release of all previously acquired resources by using `eventReleaseAllResources`.

Input Arguments

resource — Specify the name of resources for acquisition or release requests
character vector

Specify the name of the resources for the `eventAcquireResource` or the `eventReleaseResource` requests. You can specify more than one resource.

amount — Specify the amount of resources for acquisition or release requests
double

Specify the amount of resources for the `eventAcquireResource` or the `eventReleaseResource` requests.

Output Arguments

resRequest — Resource request
vector of MATLAB structures

Resource request for an acquisition or a release event specified as a vector of MATLAB structures.

Examples

A Simple Resource Specification Example

When an entity enters the storage element, it acquires resources. The entity acquires one resource of type `Resource1` and one resource of type `Resource2`, which are defined as specifications. The specifications are then used for `eventAcquireResource` with tag `MyResourceAcquireEvent`.

```
function [entity,events] = entry(obj, storage, entity, source)
        % On entry, acquire one resource of type Resource1 and one resource of type Resource2.
        resRequest(1) = obj.resourceSpecification('Resource1', 1);
        resRequest(2) = obj.resourceSpecification('Resource2', 1);
        events = obj.eventAcquireResource(resRequest, 'MyResourceAcquireEvent');
end
```

Resource Specification in a Custom Resource Acquirer Block

This example shows how to use resource management methods to create a custom entity storage block in which entities acquire resources from specified Resource Pool blocks.

Suppose that you manage a facility that produces parts from two different materials, material 1 and material 2, to fulfill orders. After a part is produced, it is evaluated for quality assurance.

Two testing methods for quality control are:

- Test 1 is used for parts that are produced from material 1.
- Test 2 is used for parts that are produced from material 2

After the production phase, parts are tagged based on their material to apply the correct test.

For more information, see "Create a Custom Resource Acquirer Block".

```
classdef CustomBlockAcquireResources < matlab.DiscreteEventSystem
    % Custom resource acquire block example.

    methods(Access = protected)

        function num = getNumInputsImpl(obj)
            num = 1;
        end

        function num = getNumOutputsImpl(obj)
            num = 1;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityTypes(1) = obj.entityType('Part');
        end

        function [input, output] = getEntityPortsImpl(obj)
            input  = {'Part'};
            output = {'Part'};
        end

        function [storageSpec, I, O] = getEntityStorageImpl(obj)
            storageSpec(1) = obj.queueFIFO('Part', 1);
            I = 1;
```

```
            O = 1;
        end

        function resNames = getResourceNamesImpl(obj)
            % Define the names of the resources to be acquired.
            resNames = obj.resourceType('Part', {'Test1', 'Test2'}) ;
        end

    end

    methods

        function [entity,events] = entry(obj, storage, entity, source)
            % On entity entry, acquire a resource from the specified pool.
            if entity.data.Test == 1
            % If the entity is produced from Material1, request Test1.
            resReq = obj.resourceSpecification('Test1', 1);
            else
            % If the entity is produced from Material2, request Test2.
            resReq = obj.resourceSpecification('Test2', 1);
            end
            % Acquire the resource from the corresponding pool.
            events = obj.eventAcquireResource(resReq, 'TestTag');
        end

        function [entity,events] = resourceAcquired(obj, storage,...
                                    entity, resources, tag)
        % After the resource acquisition, forward the entity to the output.
            events = obj.eventForward('output', storage, 0.0);
        end

    end

end
```

See Also

cancelAcquireResource | eventAcquireResource | eventForward | eventReleaseResource
| getResourceNamesImpl | matlab.DiscreteEventSystem | resourceAcquired

Topics
"Create a Custom Resource Acquirer Block"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2019a

resourceType

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Specify an entity type and the name of the resources to be acquired by the specified entity

Syntax

```
resType = resourceType(entityType,resourceNames)
```

Description

`resType = resourceType(entityType,resourceNames)` specifies an entity type and the corresponding resources that this entity type acquires.

Input Arguments

`entityType` — Name of the entity
character vector

The entity type name used in a discrete-event system. For more information, see `getEntityTypesImpl`.

`resourceNames` — Name of the resource the entity acquires
character vector

Array of resources from which the system intends to acquire resources for the defined entity type.

Output Arguments

`resType` — Resource Type
vector of MATLAB structures

Resource types returned as a vector.

Examples

Use this method together with `getResourceNamesImpl` to specify the resources of types `Test1` and `Test2` to be acquired by the entity type `Part`.

```
function resNames = getResourceNamesImpl(obj)
    % Define the names of the resources to be acquired.
    resType = obj.resourceType('Part', {'Test1', 'Test2'}) ;
end
```

See Also
cancelAcquireResource | eventAcquireResource | eventForward | eventReleaseResource | getResourceNamesImpl | matlab.DiscreteEventSystem | resourceAcquired

Topics
"Create a Custom Resource Acquirer Block"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2019a

setupEvents

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Initialize entity generation events

Syntax

```
events=setupEvents(obj)
[events,out1,...]=setupEvents(obj)
```

Description

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

`[events,out1,...]=setupEvents(obj)` specifies such event actions of the object when the block has one or more signal output ports.

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.

`out1` — Data output
any value

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

Examples

Schedule Two Entity Generation Events

Schedules two entity generation events at the start of the simulation

```
function events = setupEvents(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
```

See Also

blocked | entry | exit | generate | iterate | matlab.DiscreteEventSystem | timer

Topics

"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

testEntry

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Event action to accept or refuse entity

Syntax

```
accept = testEntry(obj,storage,entity,source,in1,...)
```

Description

`accept = testEntry(obj,storage,entity,source,in1,...)` specifies if the storage can accept entity.

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. Entity has these fields:

- `sys` (MATLAB `structure`) — It has these fields:
  - `id` (double) — Entity ID
  - `priority` (double) — Entity priority
- `data` — Entity data

`source` — Source location
double

Source location of entity, such as an input port or a storage element. It has these fields:

- `type` (character vector) — Specify `input` or `storage`
- `index` (double) — Input or storage index

`in1` — Signal input
any value

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

Output Arguments

accept — Accept or Refuse
boolean

Storage accepts entity if `true`. Otherwise, if `false`, it does not accept the entity.

Examples

Event Action to Accept or Refuse Entity

Accept or refuse an entity entering the storage.

```matlab
function bool = testEntry(obj,storage,entity,src)
    % Test if entity is accepted
    bool = obj.isEntityAcceptable(obj, entity);
    end
```

See Also
iterate | matlab.DiscreteEventSystem

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2018a

timer

Class: `matlab.DiscreteEventSystem`
Package: `matlab`

Event action when timer completes

Syntax

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

Description

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

`[entity,events,out1,...]=timer(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. Entity has these fields:

- `sys` (MATLAB `structure`) — It has these fields:
  - `id` (double) — Entity ID
  - `priority` (double) — Entity priority
- `data` — Entity data

`tag` — Tag
character vector

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. Use `matlab.DiscreteEventSystem` class methods to create events. Each event has these fields:

- `type` (`character vector`) — Type of the event
- `delay` (`double`) — Delay before the event
- `priority` (`double`) — Priority of the event
- `Storage` (`double`) — Index of the storage element
- `tag` (`character vector`) — Event tag
- `location` (MATLAB `structure`) — Source or destination location of entity

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] = timer(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
```

Custom Entity Storage Block with Timer Events

This example uses a custom entity storage block with one input, two outputs, and a storage element. An entity of type `Part` with `TimeOut` attribute enters the storage of the custom block to be processed. `TimeOut` determines the maximum allowed processing time of the parts. When a part enters the storage, two timer events are activated. One timer tracks the processing time of the part in

the oven. When this timer expires, the entity is forwarded to output 1. Another timer acts as a fail-safe and tracks if the maximum allowed processing time is exceeded or not. When this timer expires, the process is terminated and the entity is forwarded to the output 2.

For more information, see "Custom Entity Storage Block with Multiple Timer Events".

```matlab
classdef CustomEntityStorageBlockTimer < matlab.DiscreteEventSystem

    % A custom entity storage block with one input port, two output ports, and one storage.

    % Nontunable properties
    properties (Nontunable)
    % Capacity
        Capacity = 1;
    end

    methods (Access=protected)

        function num = getNumInputsImpl(~)
            num = 1;
        end

        function num = getNumOutputsImpl(~)
            num = 2;
        end

        function entityTypes = getEntityTypesImpl(obj)
            entityTypes = obj.entityType('Part');
        end

        function [inputTypes,outputTypes] = getEntityPortsImpl(obj)
            inputTypes = {'Part'};
            outputTypes = {'Part' 'Part'};
        end

        function [storageSpecs, I, O] = getEntityStorageImpl(obj)
            storageSpecs = obj.queueFIFO('Part', obj.Capacity);
            I = 1;
            O = [1 1];
        end

    end


    methods

        function [entity,event] = PartEntry(obj,storage,entity,source)
            % Specify event actions when entity enters storage.
             ProcessingTime=randi([1 15]);
             event1 = obj.eventTimer('TimeOut', entity.data.TimeOut);
             event2 = obj.eventTimer('ProcessComplete', ProcessingTime);
             event = [event1 event2];
        end

        function [entity, event] = timer(obj,storage,entity,tag)
            % Specify event actions for when scheduled timer completes.
            event = obj.initEventArray;
            switch tag
                case 'ProcessComplete'
                    event = obj.eventForward('output', 1, 0);
                case 'TimeOut'
                    event = obj.eventForward('output', 2, 0);
            end

        end

    end
end
```

See Also

blocked | destroy | entry | exit | generate | getEntityTypesImpl | iterate | matlab.DiscreteEventSystem | setupEvents

Topics
"Custom Entity Storage Block with Multiple Timer Events"
"Create Custom Blocks Using MATLAB Discrete-Event System Block"

Introduced in R2016a

simevents

Open SimEvents library

Syntax

```
simevents
```

Description

`simevents` opens the main SimEvents library.

SimEvents integrates discrete-event system modeling into the Simulink® time-based framework. In time-based systems, a signal changes value in response to the simulation clock, and state updates occur synchronously with time. In discrete-event or event-based systems, state transitions depend on asynchronous discrete incidents called events.

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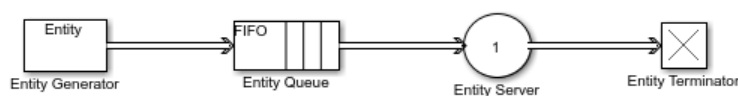
In SimEvents:

- Entity — A discrete item or object of interest based on the application domain. For example an entity can represent vehicles arriving at a gas station, messages within a communication network, planes on a runway, or trains within a signaling system.
- Event — Asynchronous discrete incidents. For example, an event can represent an entity entry to a block or entity departure from a block.
- Event Action — A custom action invoked by an event. You can customize event actions using MATLAB code that performs calculations and Simulink function calls.

SimEvents blocks can produce, process, and route entities. The blocks can also attach data to entities and manipulate entity data using event actions.

Examples

Create A Simple Queuing System

Open the SimEvents library and use the blocks from the library to build a queuing system that generates entities, queues them in a specified order, services them to change their attributes, and terminates them to represent their departure. For more information, see "A Simple Queuing System".

```
simevents;
```

Model an M/M/1 Queuing System

Open the SimEvents library and use the blocks to model a single-queue single-server system with a single traffic source and an infinite storage capacity. In the notation, the M stands for Markovian; M/M/1 means that the system has a Poisson arrival process, an exponential service time distribution, and one server. Queuing theory provides exact theoretical results for some performance measures of an M/M/1 queuing system and this model makes it easy to compare empirical results with the corresponding theoretical results. For more information, see "M/M/1 Queuing System".
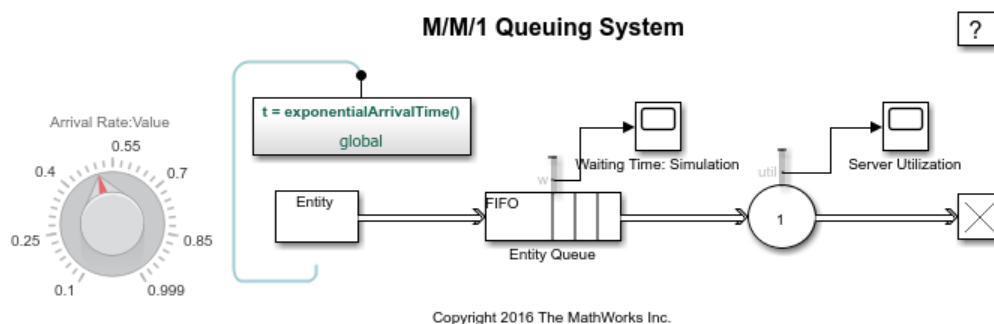

```
simevents;
```

See Also

Entity Generator | Entity Queue | Entity Server | `entityType` | `matlab.DiscreteEventSystem`

Topics
"Create a Hybrid Model with Time-Based and Event-Based Components"
"Discrete-Event Simulation in Simulink Models"
"Create a Discrete-Event Model"
"M/M/1 Queuing System"

Introduced in R2011b

simeventslib

Open legacy SimEvents library

Syntax

`simeventslib`

Description

`simeventslib` opens the version 4.4.1 of the legacy SimEvents library. not

---

Note This page is for the legacy library. To open the new SimEvents library that is introduced in R2016a, see `simevents`.

See Also

`simevents`

Topics
"Migration Considerations"

Introduced before R2006a

simevents.SimulationObserver class

Package: simevents
Superclasses: handle

Interface to create your custom observer for models with SimEvents blocks

Description

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

Class Attributes

Abstract	false
HandleCompatible	true
StrictDefaults	false

For information on class attributes, see "Class Attributes".

Creation

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

Input Arguments

modelName — Model to observe
character vector

The name of the model to observe.

Methods

Public Methods

simStarted	Specify behavior when simulation starts
simPaused	Specify behavior when simulation pauses
simResumed	Specify behavior when simulation resumes
simTerminating	Define observer behavior when simulation is terminating
getBlocksToNotify	Specify list of blocks to be notified of entity entry and exit events
notifyEventCalendarEvents	Specify whether you want notification for all events in event calendar
postEntry	Specify behavior after an entity enters a block that has entity storage
preExit	Specify behavior before an entity exits a block with entity storage
preExecute	Specify behavior before execution of an event

Protected Methods

addBlockNotification	Add block to list of blocks to be notified

removeBlockNotification	Remove block from list of blocks being notified
getEventCalendars	Get handles to event calendars
getAllBlockWithStorages	Get list of blocks that store entities
getHandleToBlock	Return block handle for a given block path
getHandlesToBlockStorages	Return storage handles of specified block

Examples

Construct Animator

This example shows how to construct an animator.

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

Create an observer to count entity departures and acquire departure timestamps

This example shows how to create a simulation observer object and use it to observe entities in a model. For more information, see "Observe Entities Using simevents.SimulationObserver Class".

Create the observer.

```
classdef myObserverPreexit < simevents.SimulationObserver
    % Add the observer properties.
    properties
    Model
    % Initialize the property count.
    count
    end
properties (Constant, Access=private)
    increment = 1;
end
methods
    % Observe any model by incorporating its name to MyObserverPreexit.
    function this = myObserverPreexit(Model)
        % Input model name to the simulation observer.
        this@simevents.SimulationObserver(Model);
        this.Model = Model;
    end
    % Initialize the count in the simulation start.
    function simStarted(this)
        this.count = 0;
    end
    % Specify list of blocks to be notified of entity entry and exit
    % events.
    function Block = getBlocksToNotify(this)
        Block = this.getAllBlockWithStorages();
    end
    function preExit(this,evSrc,Data)
        % Get the names of all storage blocks that the entities depart.
        % This returns the block with its path.
        Block = Data.Block.BlockPath;
        % Remove the path to display only the
        % block name.
        Block = regexprep(Block,'ObserverPreexitModel/' ,'');
        % Initialize the blocks to observe.
        BlockName = 'Entity Server';
        % If the block that entity exits contains the block name
        % acquire data for exit time and block name.
        if contains(Block, BlockName)
            % Get time for entity preexit from event calendar.
```

```
        evCal = this.getEventCalendars;
        Time = evCal(1).TimeNow;
        % Increase the count for departing entities.
        this.count = this.count + this.increment;
        myInfo = [' At time ',num2str(Time), ...
        ' an entity departs ', Block, ', Total entity count is ', ...
        num2str(this.count)];
        disp(myInfo);
        end
    end
end
```

Save the file as myObserverPreexit.m.

Enable the observer object to monitor ObserverPreexitModel model.

```
obj  = myObserverPreexit('ObserverPreexitModel');
```

See Also
simevents.SimulationObserver | addBlockNotification | getAllBlockWithStorages | getBlocksToNotify

Topics
"Observe Entities Using simevents.SimulationObserver Class"

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
character vector

Object of class SimulationObserver

blkPath — Full path of the block to be notified
character vector

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

See Also
getAllBlockWithStorages | getBlocksToNotify | getEventCalendars | getHandleToBlock | getHandlesToBlockStorages | notifyEventCalendarEvents | postEntry | preExecute | preExit | removeBlockNotification | simPaused | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

Object of class SimulationObserver

Output Arguments

allBlkPaths — Paths of all blocks that store entities
cell array of character vectors

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
```

See Also

addBlockNotification | getBlocksToNotify | getEventCalendars | getHandleToBlock | getHandlesToBlockStorages | notifyEventCalendarEvents | postEntry | preExecute | preExit | removeBlockNotification | simPaused | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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 preExit and postEntry methods will be called. Specify '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
character vector

Object of class SimulationObserver

Output Arguments

blks — List of blocks being notified of runtime events
{} (default) | cell array of character vectors

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
```

See Also

addBlockNotification | getAllBlockWithStorages | getEventCalendars | getHandleToBlock | getHandlesToBlockStorages | notifyEventCalendarEvents | postEntry | preExecute | preExit | removeBlockNotification | simPaused | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

Object of class SimulationObserver

Output Arguments

evCal — 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=getEventCalendars(obj);
    tNow=evcal(1).TimeNow;
    disp(tNow);
end
```

See Also

addBlockNotification | getAllBlockWithStorages | getBlocksToNotify | getHandleToBlock | getHandlesToBlockStorages | notifyEventCalendarEvents | postEntry | preExecute | preExit | removeBlockNotification | simPaused | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

Object of class SimulationObserver

`blkPath` — Full path to block
character vector

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
```

See Also

addBlockNotification | getAllBlockWithStorages | getBlocksToNotify | getEventCalendars | getHandleToBlock | notifyEventCalendarEvents | postEntry | preExecute | preExit | removeBlockNotification | simPaused | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

Object of class SimulationObserver

blkPath — Full path to block
character vector

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
```

See Also

addBlockNotification | getAllBlockWithStorages | getBlocksToNotify | getEventCalendars | getHandlesToBlockStorages | notifyEventCalendarEvents | postEntry | preExecute | preExit | removeBlockNotification | simPaused | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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) is 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 preExecute method for all events in the event calendar.

Input Arguments

obj — SimulationObserver object
character vector

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 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
```

See Also
addBlockNotification | getAllBlockWithStorages | getBlocksToNotify | getEventCalendars | getHandleToBlock | getHandlesToBlockStorages | postEntry | preExecute | preExit | removeBlockNotification | simPaused | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

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
```

See Also
addBlockNotification | getAllBlockWithStorages | getBlocksToNotify |
getEventCalendars | getHandleToBlock | getHandlesToBlockStorages |
notifyEventCalendarEvents | preExecute | preExit | removeBlockNotification |
simPaused | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

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
```

See Also

addBlockNotification | getAllBlockWithStorages | getBlocksToNotify | getEventCalendars | getHandleToBlock | getHandlesToBlockStorages | notifyEventCalendarEvents | postEntry | preExit | removeBlockNotification | simPaused | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

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);
```

```matlab
            % 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
```

See Also

addBlockNotification | getAllBlockWithStorages | getBlocksToNotify | getEventCalendars | getHandleToBlock | getHandlesToBlockStorages | notifyEventCalendarEvents | postEntry | preExecute | removeBlockNotification | simPaused | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

Object of class SimulationObserver

blkPath — Full path of the block to be notified
character vector

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

See Also

addBlockNotification | getAllBlockWithStorages | getBlocksToNotify | getEventCalendars | getHandleToBlock | getHandlesToBlockStorages | notifyEventCalendarEvents | postEntry | preExecute | preExit | simPaused | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

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
```

See Also
addBlockNotification | getAllBlockWithStorages | getBlocksToNotify | getEventCalendars | getHandleToBlock | getHandlesToBlockStorages | notifyEventCalendarEvents | postEntry | preExecute | preExit | removeBlockNotification | simResumed | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

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
```

See Also
addBlockNotification | getAllBlockWithStorages | getBlocksToNotify | getEventCalendars | getHandleToBlock | getHandlesToBlockStorages | notifyEventCalendarEvents | postEntry | preExecute | preExit | removeBlockNotification | simPaused | simStarted | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

Object of class SimulationObserver

Examples

Initialize Animation Canvas

Initialize the animation canvas.

```matlab
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
```

See Also

addBlockNotification | getAllBlockWithStorages | getBlocksToNotify | getEventCalendars | getHandleToBlock | getHandlesToBlockStorages | notifyEventCalendarEvents | postEntry | preExecute | preExit | removeBlockNotification | simPaused | simResumed | simTerminating

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

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
character vector

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
```

See Also

addBlockNotification | getAllBlockWithStorages | getBlocksToNotify |
getEventCalendars | getHandleToBlock | getHandlesToBlockStorages |
notifyEventCalendarEvents | postEntry | preExecute | preExit |
removeBlockNotification | simPaused | simResumed | simStarted

Topics
"Use SimulationObserver Class to Monitor a SimEvents Model"

Introduced in R2016a

Blocks

Composite Entity Creator

Create composite entities
Library: SimEvents

Description

The Composite Entity Creator 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.

You can combine entities from different paths using this block. The combined entity represents different parts within a larger item, such as the header, payload, and trailer that are parts of a data packet. Alternatively, you can model resource allocation by combining an entity that represents a resource with an entity that represents a part or other item.

The Composite Entity Creator block detects when all necessary component entities are present and when the composite entity that results from the combining operation will be able to advance to the next block. You can also configure the Composite Entity Creator block to make the combining operation reversible via the Composite Entity Splitter block.

Ports

Input

port_1 — Input entity
scalar | vector | matrix

Input entity port for entities entering the block.

Data Types: single | double | int8 | int16 | int32 | uint8 | uint16 | uint32 | Boolean | enumerated | bus | fixed point

port_2 — Input entity
scalar | vector | matrix

Input entity port for entities entering the block.

Data Types: single | double | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64 | Boolean | enumerated | bus | fixed point

Output

port_1 — Output composite entity
scalar | vector | matrix

Output entity port for composite entities exiting the clock.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Parameters

Number of input ports — Specify the number of input ports
2 (default) | scalar

Specify the number of input ports to create the composite entity.

Programmatic Use
Block Parameter: `NumberInputPorts`
Type: character vector
Values: `'2'` | scalar
Default: `'2'`

Entity type name — Specify the type name of the composite entity
Combined (default) | character vector

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

Programmatic Use
Block Parameter: `EntityTypeName`
Type: character vector
Values: `'Combined'` | character vector
Default: `'Combined'`

Bus object — Specify the output
off (default) | on

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

Programmatic Use
Block Parameter: `BusObject`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Define input entity names — Specify the names of the input entities used to generate the composite entity
{E1, E2} (default) | character vector

Names of the entities in the composite entity.

Programmatic Use
Block Parameter: `InputEntityName`
Type: character vector
Values: `'E1|E2'` | character vector
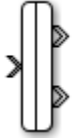**Default**: `'E1|E2'`

See Also
Composite Entity Splitter | Entity Gate | Entity Generator | Entity Input Switch | Entity Output Switch | Entity Queue | Entity Server | Entity Terminator | Resource Acquirer | Resource Pool | Resource Releaser

Introduced in R2016a

Composite Entity Splitter

Split composite entities
Library: SimEvents

Description

The Composite Entity Splitter block splits a composite entity into its individual entities and outputs them through each unblocked entity output port. A composite entity can be the output of the Composite Entity Creator block, a structured type or a bus type entity from Entity Generator or Message Send blocks.

Ports

Input

port_1 — Input composite entity
scalar | vector | matrix

Input entity port for composite entities entering the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Output

port_1 — Output entity
scalar | vector | matrix

Output port for entities exiting the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

port_2 — Output entity
scalar | vector | matrix

Output port for entities exiting the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32` | `Boolean` | `enumerated` | `bus` | `fixed point`

Parameters

Number of output ports — Specify the number of output ports
2 (default) | scalar

Specify the number of output ports to output entities.

Programmatic Use
Block Parameter: NumberOutputPorts
Type: character vector
Values: '2' | scalar
Default: '2'

See Also

Composite Entity Creator | Entity Generator
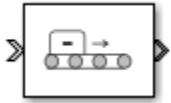
Topics
"SimEvents Common Design Patterns"

Introduced in R2016a

Conveyor System

Transport entities
**Library:**   SimEvents

Description

The Conveyor System block transports entities across the surface of a conveyor. Entities enter the block by sliding onto the conveyor surface and they depart the block by sliding off. You can specify the speed of the conveyor. If the conveyor speed is variable, a second input port appears on the block to accept anonymous entities that carry data for specifying the new conveyor speed. Use this block to model transportation applications involving production systems, or logistical systems.

In the Conveyor System block:

* You can specify the speed and the surface length of the conveyor. You can specify the length and the minimum distance between the transported entities.
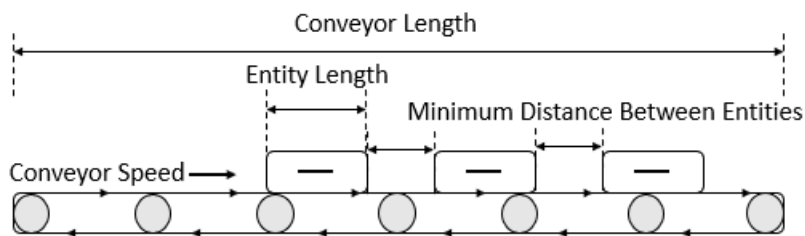
* Entities slide into the conveyor surface and an entity is considered as inside the conveyor surface when its front side coincides with the surface entry.
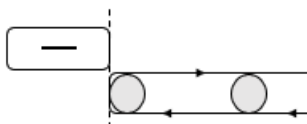

* Entities slide out of the conveyor surface and an entity is considered as outside the conveyor surface when its back side coincides with the surface entry.
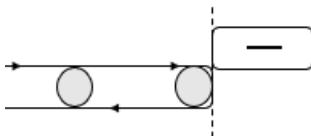
* Conveyor speed determines the total time between an entity entry to the surface and its exit from the surface.

    Entity can enter and exit the conveyor surface provided that:

- There are no other entities blocking the entity when the **Blocked output behavior** is set to `Accumulate`.
- The conveyor surface is not paused when the **Blocked output behavior** is set to `Pause`. For more information, see "Blocked output behavior" on page 2-0 .
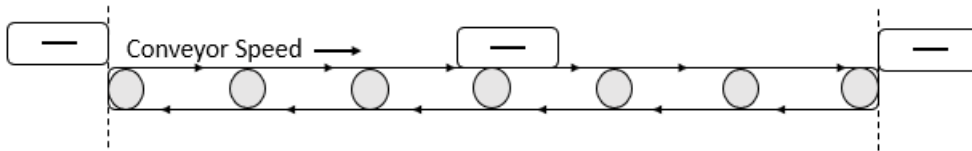

For instance, suppose that the conveyor length is `10`, entity length is `1`, and conveyor speed is `5`. Then, it takes `2.2` simulation time for the entity to depart the surface. `10/5 = 2` to travel through the surface and `1/5 = 0.2` to depart from the surface because its length is `1`.

- After an entity enters the conveyor surface, the next entity enters after the first one travels to the specified minimum distance between entities.
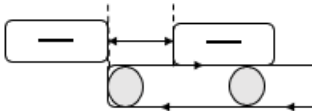
- The capacity of the conveyor system is the maximum number of entities allowed on the surface. The capacity is determined by the total surface length, entity length, and the minimum distance between entities.

  For example, suppose that the entity length is `1` meter, conveyor system surface length is `100` meters, and the distance between entities is `15` meters. The capacity of the conveyor system becomes `6` entities.
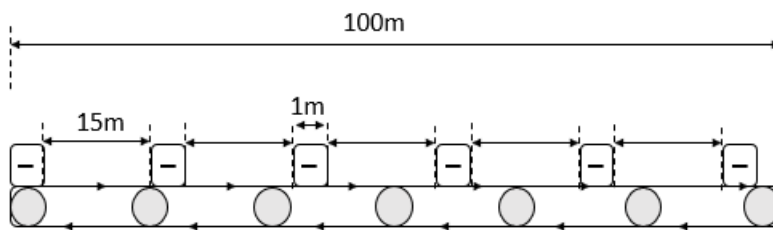
- When using the **Conveyor length**, **Conveyor speed**, **Minimum distance between entities**, and **Entity length value** parameters ensure that the values are consistent with each other.

  For example, specify the entity length of 10 cm and a conveyor system of length 100 m to achieve consistency:

  - Set the **Conveyor length** parameter to `100`.
  - Set the **Entity length** parameter to 0.1.

Ports

Input

Port_1 — Incoming entity
scalar | vector | matrix

Input entity port for entities entering the queue. Entities are not accepted by the block when the speed of the conveyor is `0`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

S — Incoming anonymous entity
scalar

Input port for anonymous entities entering the block. The entity carries data that specifies the new conveyor speed upon its arrival. The entity data must be a non-negative value between `0` and `Inf`. The new speed applies to all entities in the block including the incoming and existing ones.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Output

Port_1 — Exiting entity
scalar | vector | matrix

Output entity port for entities leaving the queue

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_d — Number of entities departed
off (default) | on

Selecting this check box outputs the number of entities that have exited the block.

Dependencies

To enable this port, select the **Statistics > Number of entities departed, d** check box.

Data Types: `double`

Port_n — Number of entities in block
off (default) | on

Selecting this check box outputs the number of entities in the block.

Dependencies

To enable this block, select the **Statistics > Number of entities in block, n** check box.

Data Types: `double`

Port_pe — Pending entity in block
off (default) | on

Selecting this check box outputs the value `1` for a pending entity in the block, and `0` otherwise.

Dependencies

To enable this port, select the **Statistics > Pending entity in block, pe**.

Data Types: `double`

Port_s — Conveyor speed
off (default) | on

Selecting this check box outputs the conveyor speed.

Dependencies

To enable this port, select the **Statistics > Conveyor speed, s**.

Data Types: `double`

Parameters

Conveyor length — Length of surface
100 (default) | numeric

Length of surface that entities travel on. For more information, see "Description" on page 2-7.

Programmatic Use
Block Parameter: `mConveyorLength`
Type: character vector
Values: `'100'` | scalar
Default: `'100'`

Conveyor speed — Speed of the conveyor surface
1000 (default) | numeric

Speed of surface that entities travel on. The speed can take values greater than 0 and its value can be zero only when the **Variable conveyor speed** check box is selected. For more information, see "Description" on page 2-7.

Programmatic Use
Block Parameter: `mConveyorSpeed`
Type: character vector
Values: `'1000'` | scalar
Default: `'1000'`

Minimum distance between entities — Minimum physical separation
0 (default) | numeric

Minimum physical separation entities maintain while moving across the conveyor system. For more information, see "Description" on page 2-7.

Programmatic Use
Block Parameter: `mMinDisBetEntity`
Type: character vector
Values: `'0'` | scalar
Default: `'0'`

Entity length source — Entity length source
Dialog (default) | Attribute

Provide entity length, selected from the drop-down list.

Dependencies

- `Dialog` — Selecting this option enables the **Entity length value** parameter.
- `Attribute` — Selecting this option enables the **Entity length attribute name** parameter.

Programmatic Use
Block Parameter: mEntityLengthSource
Type: character vector
Values: 'Dialog' | 'Attribute'
Default: 'Dialog'

Entity length value — Length of entities
1 (default) | numeric

Length of entities, specified as a numeric value. For more information, see "Description" on page 2-7.

Dependencies

To enable this parameter, select `Dialog` for **Entity length source**.

Programmatic Use
Block Parameter: mEntityLength
Type: character vector
Values: '1' | scalar
Default: '1'

Entity length attribute name — Name of entity length attribute
Length (default) | character vector

Name of entity length attribute, specified as a character vector.

Dependencies

To enable this parameter, select `Attribute` for **Entity length source**.

Programmatic Use
Block Parameter: mEntityLengthAttrName
Type: character vector
Values: 'Length' | character vector
Default: 'Length'

Blocked output behavior — Behavior when the output is blocked
Accumulate (default) | Pause | Error

Behavior when output is blocked, selected from drop-down list:

- `Accumulate` — Accumulate entities. In `Accumulate` mode, an entity continues to move on the conveyor surface until its movement is blocked by another entity ahead.

  For example, if one or more entities are extracted out of the conveyor surface by the Entity Find block, entities that are behind the extracted entity continue to move forward until they occupy the empty space due to the extraction.

- `Pause` — Pause conveyor system. In `Pause` mode, all entities on the conveyor surface stop and move together. The conveyor surface stops moving when an entity at the exit is unable to depart. During the pause, the conveyor system does not accept new entities.

- `Error` — Return an error. In `Error` mode, when an entity is blocked from entering the conveyor surface the error is displayed.

Programmatic Use
Block Parameter: mOutputBlockedOpt
Type: character vector
Values: 'Accumulate' | 'Pause' | 'Error'
Default: 'Accumulate'

Error if conveyor full — Conveyor behavior when the maximum number of entities is reached
on (default) | off

Conveyor behavior when the conveyor surface is full.

☑ on

   Return an error if the conveyor system is full.

☐ off

   Do not return an error if the conveyor system is full.

Programmatic Use
Block Parameter: mErrorUponFullOpt
Type: character vector
Values: 'on' | 'off'
Default: 'on'

Variable conveyor speed — Constant or variable conveyor speed
off (default) | on

Specify if the conveyor system has constant or variable speed.

☐ off

   Constant speed conveyor system. You can specify the speed using the **Conveyor speed** parameter.

☑ on

   Variable speed conveyor system. An input port appears to accept anonymous entities which carry data to specify the new conveyor speed.

Programmatic Use
Block Parameter: mIsVariableSpeed
Type: character vector
Values: 'on' | 'off'
Default: 'on'

Number of entities departed, d — Number of entities departed
off (default) | on

Number of entities that have departed the block.

Programmatic Use
Block Parameter: mNumEntitiesDepOpt

Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Number of entities in block, n — Number of existing entities
off (default) | on

Outputs the number of entities present in the block.

Programmatic Use
Block Parameter: `mNumEntitiesInBlockOpt`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Pending entity in block, pe — Pending entities
off (default) | on

Indicates whether an entity that is yet to depart is present in the block. The value is 1 for a pending entity, and 0 otherwise.

Programmatic Use
Block Parameter: `mEntityPendingOpt`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Number of entities extracted, ex — Number of entities extracted from this block
off (default) | on

Outputs the number of extracted entities which are pulled out from this block by the Entity Find block. When an entity is extracted, **Number of entities departed, d**, and **Number of entities in block, n** statistics are updated accordingly. For more information about finding and extracting entities, see "Find and Extract Entities in SimEvents Models".

Programmatic Use
Block Parameter: `mNumExtractedFromBlockOpt`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Conveyor speed, s — Speed of the conveyor for the duration of the simulation
off (default) | on

Outputs the speed of the conveyor during the course of the simulation.

Programmatic Use
Block Parameter: `mConveyorSpeedOpt`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

See Also

Entity Store | Entity Queue | Entity Server | MATLAB Discrete-Event System | MATLAB System

Topics
"Overview of Queues and Servers in Discrete-Event Simulation"

Introduced in R2017b

Discrete-Event Chart

Discrete event chart
Library: SimEvents

Description

The Discrete Event Chart block is similar to a Stateflow® chart but is used for discrete events. The block requires a Stateflow license.

The distinguishing characteristic of the Discrete Event Chart block is that it executes in an event-based rather than time-based fashion. The Discrete Event Chart block provides these advantages for discrete-event modeling:

- Precise timing — The time resolution for occurrence of events can be arbitrarily precise and is not limited by the sample time of the model.
- Trigger on arrival — A Discrete Event Chart block executes immediately on message arrival. It does not wait for the next sample time hit.
- Variable execution order — A Discrete Event Chart block does not have a fixed sorted execution order. The order of execution depends on the run-time conditions of the model.
- Multiple executions per time step — A Discrete Event Chart block can execute zero or multiple times in a single time step.

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

To access the chart properties, right-click the chart and select **Properties**. For more information about the block properties, see "Create Custom Queuing Systems Using Discrete-Event Stateflow Charts".

For information about SimEvents common design patterns with Discrete Event Chart block, see "SimEvents Common Design Patterns".

See Also
Entity Generator | MATLAB Discrete-Event System

Topics
"Create Custom Queuing Systems Using Discrete-Event Stateflow Charts"
"SimEvents Common Design Patterns"

Introduced in R2016a

Entity Batch Creator

Create batch of entities
Library: SimEvents

Description

The Entity Batch Creator receives the expected number of entities and creates a batch entity that contains all these entities. The batched entity is an array of entities. Any acquired resources have to be released using a Resource Releaser block before batching an input entity.

To customize actions when entities enter, exit, or are batched or blocked, enter MATLAB code in the `Entry action`, `Exit action`, `Batch generate action`, or `Blocked action` field of the **Event actions** tab. For more information about event actions, see "Events and Event Actions".

You can write MATLAB code to manipulate the attributes of the batched entity. For example, to access attributes after an entity batch is generated, select `Batch generate action` and use the code.

```
entity.batch
```

If the number of entities in a batch is 4, then *entity.batch* is a 4-by-1 structure array. To manipulate `Attribute1` of the third entity in the batch enter the code.

```
entity.batch(3).Attribute1
```

You can reference batched entity attributes in event actions. You cannot reference them in:

- Priority queues — Do not set **Priority source** parameter to `PriorityAttribute`.
- Entity Server block — Do not set **Service time source** parameter to `Attribute`.
- Output Switch block — Do not set **Switching criterion** parameter to `From attribute`.

To output the batch as a bus object, select the **Bus object** parameter. Consider creating a bus object for the batched entity when:

- Sending or receiving a batched entity to or from a MATLAB Discrete-Event System block.
- Sending or receiving a batched entity to or from a Discrete-Event Chart block.
- When passing full entity data to a Simulink Function block.
- When converting a batched entity to a signal using the Message Receive block.

Ports

Input

Port_1 — Input entity
scalar | vector | matrix

Input entity port for entities entering the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Output

Port_1 — Output batch entity
scalar | vector | matrix

Output entity port for batch entities exiting the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_a — Number of entities arrived
scalar

Number of entities that have arrived at the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities arrived, a**.

Data Types: `double`

Port_d — Number of entities that have departed the block
scalar

Number of entities that have departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities departed, d**.

Data Types: `double`

Port_rem — Number of entities remaining for the next batch
scalar

Number of entities that remain for the next batch.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities remaining for the next batch, rem**.

Data Types: `double`

Port_pe — Pending entity in block
off (default) | on

Selecting this check box outputs 1 for a pending entity in the block, and 0 otherwise.

Dependencies

To enable this port, select the **Statistics > Pending entity in block, pe**.

Data Types: `double`

Parameters

`Number of entities in batch` — Number of entities in one batch
4 (default) | scalar

Specify the number of entities in a batch.

Programmatic Use
Block Parameter: NumberOfEntitiesInBatch
Type: character vector
Values: '4' | scalar
Default: '4'

`Entity type name` — Name of the batched entity that is created after combining incoming entities
`Batch` (default) | character vector

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

Programmatic Use
Block Parameter: EntityTypeName
Type: character vector
Values: 'Batch' | character vector
Default: 'Batch'

`Bus object` — Specify whether to output the batched entity as a bus object
off (default) | on

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

Programmatic Use
Block Parameter: BusObject
Type: character vector
Values: 'on' | 'off'
Default: 'off'

`Input entity name` — Specify names to be attached to the input entities
batch (default) | character vector

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

Programmatic Use
Block Parameter: InputEntityName
Type: character vector
Values: 'batch' | character vector
Default: 'batch'

`Event actions` — Specify the event action
Entry (default) | Batch generate | Exit | Blocked

Specify the behavior of the entity on certain events. For example, the **Entry** action is called when the entity enters the block. To customize actions when entities enter, exit, or are batched or blocked, enter MATLAB code in the Entry action, Exit action, Batch generate action, or Blocked action field of the **Event actions** tab. For more information about event actions, see "Events and Event Actions". For an example, see "Manage Entities Using Event Actions"

Programmatic Use
Block Parameter: EntryAction, BatchGenerateAction, ExitAction, BlockedAction
Type: character vector
Values: MATLAB code
Default: ' '

Number of entities arrived, a — Outputs the number of entities that have arrived at the block
off (default) | on

Number of entities that have arrived at the block.

Programmatic Use
Block Parameter: NumberOfEntitiesArrived
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities departed, d — Outputs the number of entities departed the block
off (default) | on

Number of entities that have departed the block.

Programmatic Use
Block Parameter: NumberOfEntitiesDeparted
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities remaining for next batch, rem — Outputs the number of entities remaining for the next batch
off (default) | on

Outputs the number of entities still in the block for the next batch of entities.

Programmatic Use
Block Parameter: NumberOfEntitiesRequiredForNextBatch
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Pending entity in block, pe — Pending entities
off (default) | on

Indicates whether an entity that is yet to depart is present in the block. The value is 1 for a pending entity, and 0 otherwise.

Programmatic Use
Block Parameter: PendingEntity
Type: character vector
Values: 'on' | 'off'
Default: 'off'

See Also

Entity Batch Splitter | Entity Generator

Introduced in R2016b

Entity Batch Splitter

Split batch entities
Library: SimEvents

Description

The Entity Batch Splitter block splits a batched entity into its individual entities and outputs each entity through the output port. A batched entity is the output of the Entity Batch Creator block.

To customize actions when entities enter, exit, and are blocked or unbatched, enter MATLAB code in the `Entry action`, `Exit action`, `Blocked action`, or `Unbatch action` field of the **Event actions** tab.

Ports

Input

port_1 — Input batch entity
scalar | vector | matrix

Input entity port for entities entering the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Output

port_1 — Output entity
scalar | vector | matrix

Output entity port for entities exiting the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

port_a — Number of entities arrived
scalar

Number of entities that have arrived at the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities arrived, a**.

Data Types: `double`

port_d — Number of entities that have departed the block
scalar

Number of entities that have departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities departed, d**.

Data Types: `double`

port_rem — Number of entities remaining in the block
scalar

Number of entities still in the block that have yet to depart.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities remaining to depart, rem**.

Data Types: `double`

port_pe — Pending entity in block
off (default) | on

Selecting this check box outputs the value `1` for a pending entity in the block, and `0` otherwise.

Dependencies

To enable this port, select the **Statistics > Pending entity in block, pe**.

Data Types: `double`

Parameters

Event actions — Specify the event action
`Entry` (default) | `Unbatch` | `Exit` | `Blocked`

Specify the behavior of the entity on certain events. For example, the **Entry** action is called when the entity enters the block. To customize actions when entities enter, exit, or are unbatched or blocked, enter MATLAB code in the `Entry action`, `Exit action`, `Unbatch action`, or `Blocked action` field of the **Event actions** tab. For more information about event actions, see "Events and Event Actions". For an example, see "Manage Entities Using Event Actions"

Programmatic Use
Block Parameter: EntryAction, UnbatchAction, ExitAction, BlockedAction
Type: character vector
Values: MATLAB code
Default: `''`

Number of entities arrived, a — Outputs the number of entities that have arrived at the block
off (default) | on

Number of entities that have arrived at the block.

Programmatic Use
Block Parameter: NumberOfEntitiesArrived

Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

`Number of entities departed, d` — Outputs the number of entities that have departed the block
off (default) | on

Selecting this check box outputs the number of entities that have exited the block.

Programmatic Use
Block Parameter: `NumberOfEntitiesDeparted`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

`Number of entities remaining to depart, rem` — Outputs the number of entities that remain in the block
off (default) | on

Outputs the number of entities still in the block that have yet to depart.

Programmatic Use
Block Parameter: NumberOfEntitiesWaitingToDepart
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

`Pending entity in block, pe` — Pending entities
off (default) | on

Indicates whether an entity that is yet to depart is present in the block. The value is 1 for a pending entity, and 0 otherwise.

Programmatic Use
Block Parameter: PendingEntity
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

See Also
Entity Batch Creator | Entity Generator

Introduced in R2016b

Entity Find

Find entities
Library: SimEvents

Resource1

Description

The Entity Find block finds entities that use a specific resource. The block receives a trigger entity from its input port. Upon receiving the trigger entity, it initiates a search across all blocks in a model for entities with a specific resource. You can further filter the search for finding entities by adding additional filtering conditions. The block can manipulate the found entities in these ways.

- Search entities that satisfy a specific condition across blocks in a model to find and examine them.
- Extract found entities from the model without modifying them.
- Change the attributes of the found entities at their location in the model without extraction.

  In this case, to customize actions when the entity is found, in the **Event actions** tab, in the `On found action` field enter MATLAB code.

- Extract and change entity attributes. The extracted entities are queued in the block and rerouted through its output port.

  In this case, to customize actions when entities enter, exit, or are blocked, enter MATLAB code in the `Entry action`, `Exit action`, or `Blocked action` field of the **Event actions** tab. For more information, see "Write Event Actions for Legacy Models".

The block can extract entities from Entity Server, Entity Queue, Entity Store, Resource Acquirer, Entity Replicator, Conveyor System, Entity Selector, MATLAB Discrete-Event System, and Discrete-Event Chart blocks.

When an entity is extracted, pending events and statistics are updated accordingly. For instance, if an entity is extracted by an Entity Find block from an Entity Server block during the service period, the rest of the service is canceled and the output of the statistics are updated.

Note The block can find only one entity type that you specify in the model and it cannot find or extract entities across model reference boundaries.

For more information about common workflows involving Entity Find block, see "Find and Extract Entities in SimEvents Models".

Ports

Input

`Port_1` — Input entities to trigger finding entities
scalar | vector | matrix

Input port for entities to trigger the event of finding entities using a specific resource.

Data Types: `single | double | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64 | Boolean | enumerated | bus | fixed point`

Output

`Port_1` — Output found entity
scalar | vector | matrix

Output entity port for entities that are found by the block.

Dependencies

To enable this port, select the **Extract found entities** checkbox.

Data Types: `single | double | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64 | Boolean | enumerated | bus | fixed point`

`Port_f` — Number of entities found
scalar

Number of entities that are found by the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities found, f**.

Data Types: `double`

`Port_d` — Number of entities that have departed the block
scalar

Number of entities that have departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities departed, d**.

Data Types: `double`

`Port_n` — Number of found entities that have not yet departed the block
scalar

Number of found entities that have not yet departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of found entities in block, n**.

2-25

Data Types: `double`

Port_w — Average wait time for found entities in the block
scalar

Average wait time for found entities in the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Average wait, w**.

Data Types: `double`

Port_l — Average length of the found entity queue
scalar

Outputs the average length of the found entity queue.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Average queue length, l**.

Data Types: `double`

Port_ex — Number of entities extracted
scalar

Number of entities that are extracted out of this block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities extracted, ex**.

Data Types: `double`

Parameters

Main

Resource — Specify the name of the reference resource the entities use
`Resource1` (default) | character vector

Name of the reference resource that is used by the entities to be found by the block.

Programmatic Use
Block Parameter: `ResourceName`
Type: character vector
Values: resource name
Default: `'Resource1'`

Extract found entities — Extract and reroute found entities
`off` (default) | on

Extract and output the entities that are found in the search.

Programmatic Use
Block Parameter: EnableOutput
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Additional filtering condition — Add optional filtering condition
off (default) | on

Enable the option to define additional matching conditions for finding entities. If selected, a MATLAB editor appears where you can define additional matching conditions.

Programmatic Use
Block Parameter: EntityFilter
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Matching condition — Define the matching condition
match = true; (default)

Use MATLAB code to define additional matching conditions for finding entities. The block searches for entities such that the value of the boolean variable match becomes true. For instance, when

```
match = isequal(trigger.Attribute1, entity.Attribute1);
```

the block finds entities that have the same Attribute1 value with the trigger entity, because only their equality sets the match as true.

While authoring additional matching conditions, you can use these variables.

- trigger — Use to access the trigger entity attributes.
- entity — Use to access the attributes of the entity that is being found.
- match — Use as a Boolean value to be returned by the matching condition. The value of match is initialized as false.

Dependencies

This check box appears if the **Additional filtering condition** check box in the **Main** tab is selected.

Programmatic Use
Block Parameter: MatchingCondition
Type: character vector
Values: MATLAB code
Default: '% match = isequal(trigger.Attribute1, entity.Attribute1);match = true;'

Event Actions

Event actions — Specify the event action
OnFound (default) | Entry | Exit | Blocked

Specify the behavior of the entity on certain events. For example, the **Entry** action is called when the entity enters the block. To customize actions when entities enter, exit, or are found or blocked, enter MATLAB code in the Entry action, Exit action, On found action, or Blocked action field

of the **Event actions** tab. For more information about event actions, see "Events and Event Actions". For an example, see "Manage Entities Using Event Actions"

Dependencies

`On found action` appears only if the **Extract found entities** check box is not selected.

`Entry action`, `Exit action`, and `Blocked action` appear only if the **Extract found entities** check box is selected.

Programmatic Use
Block Parameter: `EntryAction, ExitAction, BlockedAction`
Type: character vector
Values: MATLAB code
Default: `''`

Statistics

Number of entities found, f — Outputs the number of entities that are found
off (default) | on

Number of entities that are found by the block during simulation.

Programmatic Use
Block Parameter: `NumberEntitiesDeparted`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Number of entities departed, d — Outputs the number of entities that have departed the block
off (default) | on

Number of entities that have departed the block.

Dependencies

This check box appears if the **Extract found entities** check box in the **Main** tab is selected.

Programmatic Use
Block Parameter: `NumberEntitiesDeparted`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Number of found entities in block, n — Outputs the number of found entities that are yet to depart
off (default) | on

Number of found entities that are yet to depart.

Dependencies

This check box appears if the **Extract found entities** check box in the **Main** tab is selected.

Programmatic Use
Block Parameter: `NumberEntitiesInBlock`

Type: character vector
Values: 'on' | 'off'
Default: 'off'

`Average wait, w` — Outputs the average wait time
off (default) | on

Sum of the wait times for entities departing the block divided by their total number. Wait time is the duration between the Entity Find block entry and exit of an entity. For more information, see "Interpret SimEvents Models Using Statistical Analysis".

Dependencies

This check box appears if the **Extract found entities** check box in the **Main** tab is selected.

Programmatic Use
Block Parameter: AverageWait
Type: character vector
Values: 'on' | 'off'
Default: 'off'

`Average queue length, l` — Outputs the average length of the entity queue
off (default) | on

Accumulated time-weighted average queue size. The block computes this value by:

1 Multiplying the size of the queue by its duration to calculate time-weighted queue size

2 Summing up all time-weighted queue sizes and averaging them over total time

For more information, see "Interpret SimEvents Models Using Statistical Analysis".

Dependencies

This check box appears if the **Extract found entities** check box in the **Main** tab is selected.

Programmatic Use
Block Parameter: AverageStoreSize
Type: character vector
Values: 'on' | 'off'
Default: 'off'

`Number of entities extracted, ex` — Number of entities extracted from this block
off (default) | on

Outputs the number of extracted entities.

Dependencies

This check box appears if the **Extract found entities** check box in the **Main** tab is selected.

Programmatic Use
Block Parameter: NumEntitiesExtracted
Type: character vector
Values: 'on' | 'off'
Default: 'off'

See Also

Resource Acquirer | Resource Pool | Resource Releaser
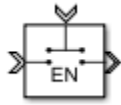
Topics
"Model Using Resources"
"SimEvents Common Design Patterns"

Introduced in R2018b

Entity Gate

Gate entities
Library: SimEvents

Description

The Entity Gate block controls when pending entities can advance in the model.

The **Operating mode** parameter specifies how the pending entities advance through the gate.

- `Enable gate` — opens and allows entities to advance whenever the control port receives an anonymous entity with a positive value, and closes whenever it has zero or a negative value. For more information, see "Use Queue Event Actions to Model a Storage Tank".

- `Release gate` — allows one pending entity to advance for each anonymous entity or message that arrives on the control port. At all other times, the entity input port of the block is unavailable.

- `Selection gate` — allows entities to advance whenever the anonymous entity value from the control port matches the attributes of the pending entities.

Use the Entity Gate block to control the flow of entities on the entity path. Use the Entity Output Switch block to select an output port for the departure of an entity among multiple entity output ports. For more information, see "Route Vehicles Using an Entity Output Switch Block".

Ports

Input

Port_1 — Input entity
scalar | vector | matrix

Input entity that carries scalar, bus, or vector data to enter the gate.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `string` | `fixed point`

control — Incoming control entity
scalar

Input control port to accept the entity that determines the state of the gate.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `fixed point`

Output

Port_1 — Exiting entity
scalar

Output entity port for entities leaving the gate.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `string` | `fixed point`

Parameters

`Operating mode` — **Select the mode of operation**
`Enable gate` (default) | `Release gate` | `Selection gate`

Select the mode of operation of this gate. With the **Operating mode** parameter set to:

- `Enable gate`, this block represents a gate that opens whenever the control port receives an anonymous entity with a positive value, and closes whenever it has zero or a negative value. By definition, an open gate permits entity arrivals as long as the entities are able to immediately advance to the next block, while a closed gate forbids entity arrivals. The anonymous entity that is received at the control port has a numerical value of type double. Since the gate receives an anonymous entity with a positive value and opens, an enabled gate remains open until it receives an entity with zero or a negative value and closes.

- `Release gate`, this block permits the arrival of one pending entity for each anonymous entity or message that arrives on the control port. 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 immediately advance to the next block.

- `Selection gate`, this block permits the arrival of pending entities whenever the anonymous entity value from the control port matches the attributes of the pending entities. Otherwise it prevents the arrival of pending entities.

Programmatic Use
Block Parameter: `OperatingMode`
Type: character vector
Values: `'Enable gate'` | `'Release gate'` `'Selection gate'`
Default: `'Enable gate'`

`Matching attribute` — **Specify name of the attribute that matches the value from the control port**
`Attribute1` (default) | character vector

Name of the attribute to match the value from the control port.

Dependencies

This parameter is visible when you set **Operating mode** to `Selection Gate`.

Programmatic Use
Block Parameter: `MatchingAttributeName`
Type: character vector
Values: `'Attribute1'` | character vector
Default: `'Attribute1'`

`Initial value from the control port` — **Specify initial value from the control port to match the matching attribute**
`NaN` (default) | scalar

Specify the initial value to match the matching attribute which opens the gate.

Dependencies

This parameter is visible when you set **Operating mode** to `Selection Gate`.

Programmatic Use
Block Parameter: `InitialValueOfMatchingAttribute`
Type: character vector
Values: `'NaN'` | scalar
Default: `'NaN'`

Open gate at simulation start — Opening the gate at the start
off (default) | on

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

☐ off

    Gate is closed at the start of the simulation.

☑ on

    Gate is open at the start of the simulation.

Programmatic Use
Block Parameter: `OpenGateAtSimulationStar`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

See Also
Composite Entity Creator | Composite Entity Splitter | Discrete Event Chart | Entity Input Switch | Entity Multicast | Entity Output Switch | Entity Queue | Entity Replicator | Entity Server
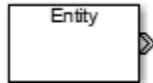
Topics
"SimEvents Common Design Patterns"

Introduced in R2016a

Entity Generator

Generate Entities
Library: SimEvents

Description

The Entity Generator block generates entities. Entities are discrete items of interest that you can define in a discrete-event simulation. An entity can carry scalar, bus, or vector data. The meaning of an entity depends on the model. Entity can represent customers in a queuing system, data packets from a remote controller to an actuator, or any discrete item you define.

By default the block entity generation method is `Time-based`. In this method, the block generates entities using intergeneration times specified by the **Period**, from an input signal or statistical distribution. See "Entities in a SimEvents Model", for more information about creating time-based and randomized entities.

The block also creates event-based entities. Choose `Event-based` as the **Generation Method** for an external event to specify the entity intergeneration time. For an example, see "Generate Entities When Events Occur".

To customize actions when the entity is generated or it exits the block, in the **Event actions** tab, in the `Generate action`, or `Exit action` field, enter MATLAB code. For more information, see "Events and Event Actions".

Ports

Input

Port_1 — Input to trigger entity generation upon arrival of events
scalar | vector | matrix

Dependencies

To enable this port, click the **Entity generation** tab and select `Event-based` for the **Generation method**.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_2 — Input for the signal that determines the intergeneration time value for the next entity generation
scalar | vector | matrix

Dependencies

To enable this port, click the **Entity generation** tab and select `Time-based` for the **Generation method** and `Signal port` for the **Time source**.

Data Types: `double`

Output

Port_1 — Output generated entity
scalar | vector | matrix

Output port for the generated entities departing the generator.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_d — Number of entities that have departed the block
scalar

Number of entities that have departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities departed, d**.

Data Types: `double`

Port_pe — Pending entity in block
off (default) | on

Outputs 1 for a pending entity, and 0 otherwise.

Dependencies

To enable this port, select the **Statistics > Pending entity in block, pe**.

Data Types: `double`

Port_w — Average intergeneration time
off (default) | on

Outputs the average time between generation of entities.

Dependencies

To enable this port, select the **Statistics > Average intergeneration time, w**.

Data Types: `double`

Parameters

Generation method — Select the method of entity generation
Time-based (default) | Event-based

Choose the entity generation method. 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.

Programmatic Use
Block Parameter: `GenerationMethod`
Type: character vector
Values: `'Time-based'` | `'Event-based'`

Default: 'Time-based'

`Time source` — Select the source of the intergeneration time
`Dialog` (default) | `Signal port` | `MATLAB action`

Specify the source for entity 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 represented by `dt`.

For more information about specifying intergeneration times for entities, see "Specify Intergeneration Times for Entities".

Dependencies

This parameter is visible when the **Generation method** is set to `Time-based`.

Programmatic Use
Block Parameter: TimeSource
Type: character vector
Values: 'Dialog' | 'Signal port' | 'MATLAB action'
Default: 'Dialog'

`Period` — Define the period between the generation of entities
1 (default) | scalar

Specify the time between entity intergeneration. For instance, if the **Period** is 1, the block waits 1 simulation time in between entity generations. See, "Specify Intergeneration Times for Entities" for more information.

Tunable: Yes

Dependencies

This parameter is visible when **Generation method** is set to `Time-based`.

Programmatic Use
Block Parameter: Period
Type: character vector
Values: '1' | scalar
Default: '1'

`Intergeneration time action` — Specify the time between entity generations
`dt = rand(1,1)` (default) | MATLAB code

Use MATLAB code to specify service time. `dt` specifies the time between entity generations. You can manually specify `dt` or use **Insert pattern** button to generate entities with a repeating sequence or from a distribution. The block uses this parameter every time it is ready for entity generation. For an example, see "Specify Intergeneration Times for Entities".

Dependencies

This parameter is visible when **Service time source** is set to `MATLAB action`.

Programmatic Use
Block Parameter: IntergenerationTimeAction
Type: character vector
Values: MATLAB code
Default: 'dt = rand(1,1);'

Generate entity at simulation start — Generate an entity at the start of the simulation
on (default) | off

Generates entity at the start of the simulation.

Programmatic Use
Block Parameter: GenerateEntityAtSimulationStart
Type: character vector
Values: 'on' | 'off'
Default: 'on'

Entity type — Choose the type of entity to generate
Structured (default) | Anonymous | Bus object

Choose the type of entity to generate.

- The Anonymous type has one data value associated with it.
- The Structured type (default) includes name and initial value attributes that you can specify.

  - You can attach more than one attribute to an entity by clicking +.

  - You can delete attributes by clicking x.

  - You can change the order of the attributes by clicking up and down.

  - You can convert a Structured type entity to a bus object by clicking convert

- The Bus object type lets you generate bus objects as entities.

  Click **Launch Bus Editor** to open bus editor and to generate bus objects. A bus object can be an element of another bus object which can be used to create hierarchy in the data that is attached to an entity.

For more information, see "Entities in a SimEvents Model".

Programmatic Use
Block Parameter: EntityType
Type: character vector
Values: 'Structured' | 'Anonymous' | 'Bus object'
Default: 'Structured'

Entity priority — Specify the priority of the generated entity
300 (default) | scalar

Determines the priority of the generated entity. The lower the value the higher the priority For more information, see "Working with Entity Attributes and Entity Priorities".

Programmatic Use
Block Parameter: EntityPriority
Type: character vector
Values: '300' | scalar
Default: '300'

Entity type name — Specify the name of the generated entity
Entity (default) | character vector

Determines the name of the generated entity.

Dependencies

This parameter is visible when **Entity type** is set to Bus object or Structured.

Programmatic Use
Block Parameter: EntityTypeName
Type: character vector
Values: 'Entity' | character vector
Default: 'Entity'

Data initial value — Specify the initial value of anonymous entity data
0 (default) | scalar | vector | matrix

Set the anonymous entity data initial value. This value cannot be of type int64 or fixed-point.

Dependencies

This parameter is visible when you set **Entity type** to Anonymous.

Programmatic Use
Block Parameter: DataInitialValue
Type: character vector
Values: '0' | scalar | vector | matrix
Default: '0'

Attribute Name — Define the name of the generated entity attribute
Attribute1 (default) | character vector

Define entity attribute name.

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

Dependencies

This parameter is visible when **Entity type** is set to Structured.

Programmatic Use
Block Parameter: AttributeName
Type: character vector
Values: 'Attribute1' | character vector
Default: 'Attribute1'

Attribute Initial Value — Define the generated entity attribute initial value
1 (default) | scalar

Specify the entity attribute initial value. This parameter is visible when **Entity type** is set to `Structured`. This value can not be of type fixed-point.

Programmatic Use
Block Parameter: AttributeInitialValue
Type: character vector
Values: 1 | scalar
Default: '1'

Event actions — Specify the behavior of the entity on certain events
Generate (default) | Exit

Define the behavior in the **Event action** parameter. The `Generate` action is called when an entity is generated and the `Exit` action is called just before an entity exits the block.

Programmatic Use
Block Parameter: GenerateAction, ExitAction
Type: character vector
Values: MATLAB code
Default: ''

Number of entities departed, d — Outputs the number of entities that have departed the block
off (default) | on

Number of entities that have departed the block.

Programmatic Use
Block Parameter: NumberEntitiesDeparted
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Pending entity present in block, pe — Pending entities
off (default) | on

Indicates whether an entity that is yet to depart is present in the block. The value is 1 for a pending entity, and 0 otherwise. This block can have at most one pending entity because its storage capacity is one. If there is an existing pending entity, the block does not generate another entity until the pending entity departs the block.

Programmatic Use
Block Parameter: PendingEntityInBlock
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Average intergeneration time, w — Average time between generation of entities
off (default) | on

Outputs the average time between generation of entities. **Average intergeneration time, w** is the ratio of the total generation time to the total number of generated entities.

Programmatic Use
Block Parameter: AverageIntergenerationTime
Type: character vector
Values: 'on' | 'off'
Default: 'off'

See Also

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

Topics
"Generate Multiple Entities at Time Zero"
"Specify Intergeneration Times for Entities"
"Generate Entities When Events Occur"
"SimEvents Common Design Patterns"
"Entities in a SimEvents Model"

Introduced in R2016a

Entity Input Switch

Switch input entities
Library: SimEvents

Description

A typical scenario in which you might use an input switch is when multiple sources of entities feed into a single queue, where the sequencing follows specific rules. For example, users of terminals in a time-shared computer submit jobs to a queue that feeds into the central processing unit, where an algorithm regulates access to the queue so as to prevent unfair domination by any one user.

Note If you want to merge message or entity paths and generate code for your component interface, use the Simulink Message Merge block. Message Merge block's behavior is the same as the Entity Output Switch block with **Active port selection** parameter set to `All`.

For an example, see "Generate Entities When Events Occur".

Combine Entity Paths

You can merge multiple paths into a single path using the Entity Input Switch block with the **Active port selection** parameter set to `All`. Merging entity paths does not change the entities themselves, just as merging lanes on a road does not change the vehicles that travel on it. In particular, the Entity Input Switch block does not create aggregates or batches.

Here are some scenarios in which you might combine entity paths:

- Attaching different data — Multiple entity generator blocks create entities having different values for a particular attribute. The entities then follow a merged path but might be treated differently later based on their individual attribute values.
- Merging queues — Multiple queues merge into a single queue.
- Connecting a feedback path — A feedback path enters the same queue as an ordinary path.

Sequence Simultaneous Pending Arrivals

The Entity Input Switch block does not experience any collisions, even if multiple entities attempt to arrive at the same time. The categories of behavior are as follows:

- If the entity output port is not blocked when the entities attempt to arrive, then the sequence of arrivals depends on the sequence of departure events from blocks that precede the Entity Input Switch 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.

2-41

- If pending entities are waiting to advance to the Entity Input Switch block when its entity output port changes from blocked to unblocked, then the entity input ports are notified of the change sequentially. The change from blocked to unblocked means that an entity can advance to the Entity Input Switch block.

  If at least two entities are waiting to advance to the Entity Input Switch block via distinct entity input ports, then the notification sequence is important because the first port to be notified of the change is the first to advance an entity to the Entity Input Switch block.

Select Arrival Path

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

You can also 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.

Ports

Input

Port_1 — Input entity
scalar | vector | matrix

Input entity port for entities entering the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_2 — Input entity
scalar | vector | matrix

Input entity port for entities entering the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

control — Incoming control entity
scalar

Input control port for the incoming control entity that determines the input port for the entities arriving at the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Output

Port_1 — Output entity
scalar | vector | matrix

Output entity port for entities exiting the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Parameters

`Number of input ports` — Specify the number of input ports
2 (default) | scalar

Determines how many entity input ports the block has.

Programmatic Use
Block Parameter: `NumberInputPorts`
Type: character vector
Values: `'2'` | scalar
Default: `'2'`

`Active port selection` — Specify the active input port
`All` (default) | `Switch`

Specify the active input port to allow arrival of entities at all ports or one port at a time. Select `All` to combine entity paths and allow arrival of entities at all ports. Select `Switch` to allow arrival of an entity at only one port at a time.

Programmatic Use
Block Parameter: `ActivePortSelection`
Type: character vector
Values: `'All'` | `'Switch'`
Default: `'All'`

`Switching criterion` — Specify input port switch criterion
`Round robin` (default) | `From control port` | `Equiprobable`

Select the criterion for switching between input ports.

- Select `Round robin` to select ports in a round robin fashion. Set the initial port in the **Initial port selection**.

- Select `From control port` to let the control port determine the selected port. A control port will appear to input an anonymous entity carrying data with a value greater than `0`, and smaller than or equal to the number of input ports to determine the active port.

- Select `Equiprobable` to let the block randomly select any port with equal probability. Set the **Seed** to generate a random number and to determine the active input port.

Programmatic Use
Block Parameter: `SwitchingCriterion`
Type: character vector
Values: `'Round robin'` | `'From control port'` | `'Equiprobable'`
Default: `'Round robin'`

`Initial port selection` — Specify the initial input port for entity entry
1 (default) | scalar

Specify initially which port allows arrival of an entity.

Programmatic Use
Block Parameter: InitialPortSelection
Type: character vector
Values: '1' | scalar
Default: '1'

Seed — Specify the seed for the random number generator to determine the input port
23453 (default) | scalar

Specify the seed for the random number generator to determine the input port.

Dependencies

This parameter is visible when **Switching criterion** is set to Equiprobable.

Programmatic Use
Block Parameter: Seed
Type: character vector
Values: '23453' | scalar
Default: '23453'

See Also

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

Topics
"SimEvents Common Design Patterns"

Introduced in R2016a

Entity Multicast

Send multicast entities
Library: SimEvents

Description

The Entity Multicast block broadcasts entities. An entity that arrives at the block is cloned into copies and sent with a **Multicast tag**. Each Entity Queue block with **Entity arrival source** specified as `Multicast` with the same **Multicast tag** receives copies. For more information, see "Overview of Queues and Servers in Discrete-Event Simulation".

Ports

Input

Port_1 — Input entity
scalar | vector | matrix

Input entity port for entities entering the block.

Note Event actions are not supported with string entity data type

---

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `string` | `fixed point`

Output

send — Broadcast entities
scalar | vector | matrix

Output broadcasted entities.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `string` | `fixed point`

Parameters

Multicast tag — Specify the tag with which to broadcast the entities
A (default) | character vector

Specify the tag with which to broadcast the entities. For example if the default tag A is used, each Entity Queue block with **Entity arrival source** specified as `Multicast` and the **Multicast tag** is set as A receives a copy.

Programmatic Use
Block Parameter: `MulticastTag`
Type: character vector
Values: `'A'` | character vector
Default: `'A'`

See Also

Entity Generator | Multicast Receive Queue

Introduced in R2016a

Entity Output Switch

Output entities
Library: SimEvents

Description

The Entity Output Switch block allows you to select an output port for the departure of an entity among multiple entity output ports. The selected port can change during the simulation and you can choose the criterion for switching between output ports.

Here are some scenarios in which you might use an output switch:

*   Entities advance to one of several queues based on efficiency or fairness concerns. For example, airplanes advance to one of several runways depending on queue length, or customers advance to the first available cashier out of several cashiers.

    Comparing different approaches to efficiency or fairness, by testing different rules to determine the selected output port of the output switch, might be part of your goal in simulating the system. For an example, see "Route Vehicles Using an Entity Output Switch Block".
*   Entities advance to a specific destination based on their characteristics. For example, parcels advance to one of several delivery vehicles based on the locations of the specified recipients.
*   Entities use an alternate route in case the preferred route is blocked. For example, a communications network drops a packet if the route to the transmitter is blocked and the simulation gathers statistics about dropped packets.

Alternatively, you can use the Entity Gate block to control the flow of entities on a specific entity path. For more information, see "Using Entity Priority to Sequence Departures".

Ports

Input

Port_1 — Input entity
scalar | vector | matrix

Input entity port for entities entering the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

control — Incoming control entity
scalar

Input control port for the incoming control entity that determines the output port for the entities departing from the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `fixed point`

Output

Port_1 — Output entity
scalar | vector | matrix

Output entity port for entities exiting the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_2 — Output entity
scalar | vector | matrix

Output entity port for entities exiting the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Parameters

Number of output ports — Specify the number of output ports
2 (default) | scalar

Number of output ports for entity departure.

Programmatic Use
Block Parameter: `NumberOutputPorts`
Type: character vector
Values: `'2'` | scalar
Default: `'2'`

Switching criterion — Choose the switching criterion
`First port that is not blocked` (default) | `Round robin` | `From control port` | `From attribute` | `Equiprobable`

Choose the criterion for switching between output ports.

- To output the entity to the first unblocked port, select `First port that is not blocked`.

  Assume an example where entities arriving at the Entity Output Switch block depart through the first entity output port that is not blocked, as long as at least one entity output port is not blocked. An everyday example of this approach is a single queue of people waiting for service by one of several bank tellers, cashiers, call center representatives, etc. Each person in the queue wants to advance as soon as possible to the first available service provider without preferring one over another.

- To output entities in a round robin fashion among the output ports, select `Round robin`.

- Select `From control port` to let the control port determine the selected port for entity departure. A control port will appear to input an anonymous entity carrying data with a value greater than 0 and smaller than or equal to the number of output ports to determine the active port.

- To specify an attribute that determines the output port, select `From attribute`. The attribute value is greater than `0` and smaller than or equal to the number of output ports to determine the active port.

  Consider the situation in which parcels are sorted among several delivery vehicles based on the locations of the specified recipients. If each parcel is an entity, then you can attach data to each entity to indicate the location of its recipient.

- To randomly select an output port for entity departure, select `Equiprobable`. Set the **Seed** to generate a random number and to determine the active output port.

Note The block rounds a double precision value to the nearest integer less than or equal to its value as port selection. For instance, the value `0.3` is rounded of to `0` which is not a valid value for port selection.

Programmatic Use
Block Parameter: `SwitchingCriterion`
Type: character vector
Values: `'First port that is not blocked'` | `'Round robin'` | `'From control port'` | `'From attribute'` | `'Equiprobable'`
Default: `'First port that is not blocked'`

`Initial port selection` — Specify the output port at the start of the simulation
1 (default) | scalar

Select the initial port for the entity departure.

Dependencies

This parameter is visible when **Switching criterion** is set to `Round robin` or `From control port`.

Programmatic Use
Block Parameter: `InitialPortSelection`
Type: character vector
Values: `'1'` | scalar
Default: `'1'`

`Switch Attribute name` — Specify the attribute that determines the output port
Attribute1 (default) | character vector

Specify the attribute name used to switch the output port.

Dependencies

This parameter is visible when **Switching criterion** is set to `From attribute`.

Programmatic Use
Block Parameter: `SwitchAttributeName`
Type: character vector
Values: `'Attribute1'` | character vector
Default: `'Attribute1'`

`Seed` — Specify the seed
34567 (default) | scalar

Specify the seed for the random number generator to determine the output port.

Dependencies

This parameter is visible when **Switching criterion** is set to `Equiprobable`.

Programmatic Use
Block Parameter: Seed
Type: character vector
Values: `'34567'` | scalar
Default: `'34567'`

See Also

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

Topics
"SimEvents Common Design Patterns"

Introduced in R2016a

Queue, Entity Queue

Enqueue messages and entities
Library: Simulink / Messages & Events
SimEvents

Description

This block stores entities or messages in a queue, based on the order of arrival or priority. Each element at the head of the queue departs when the downstream block is ready to accept it. The Queue block and the Entity Queue block are the same blocks with different default values for the **Overwrite the oldest element if queue is full** check box.

You can specify the capacity of the queue, and the policy when the queue is full. The block supports three different message or queue sorting policies, first-in-first out (FIFO), last-in-first out (LIFO), and priority. The priority queue can be used only when the **Overwrite the oldest element if queue is full** check box is cleared.

Ports

Input

Port_1 — Input entity or message
scalar | vector | matrix

Input entity or message that carries scalar, bus, or vector data to enter the queue.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Output

Port_1 — Output entity or message
scalar | vector | matrix

Output port that allows entities or messages at the head of the queue to depart when a downstream block is ready to accept them.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_d — Number of entities that have departed the block
scalar

Number of entities that have departed the block.

Dependencies

To enable this port, select **Overwrite the oldest element if queue is full** check box, and click the **Statistics** tab and select the box labeled **Number of entities departed, d**.

Data Types: `double`

Port_n — Number of entities that have not yet departed the block
scalar

Number of entities that have not yet departed the block.

Dependencies

To enable this port, select **Overwrite the oldest element if queue is full** check box, and click the **Statistics** tab and select the box labeled **Number of entities in block, n**.

Data Types: `double`

Port_w — Average wait time for entities in the block
scalar

Average wait time for entities in the block.

Dependencies

To enable this port, select **Overwrite the oldest element if queue is full** check box, and click the **Statistics** tab and select the box labeled **Average wait, w**.

Data Types: `double`

Port_l — Average length of the entity queue
scalar

Port_l outputs the average length of the entity queue.

Dependencies

To enable this port, select **Overwrite the oldest element if queue is full** check box, and click the **Statistics** tab and select the box labeled **Average queue length, l**.

Data Types: `double`

Port_ex — Number of entities extracted
scalar

Number of entities that are pulled out of this block.

Dependencies

To enable this port, select **Overwrite the oldest element if queue is full** check box, and click the **Statistics** tab and select the box labeled **Number of entities extracted, ex**.

Data Types: `double`

Parameters

Overwrite the oldest element if queue is full — Specify queue overwriting policy
on (default for Simulink) | off (default for SimEvents)

Select this check box to choose between two queue overwriting policies.

- If you select the check box, an incoming message overwrites the oldest if the queue is full.

This mode represents a simple message buffer that you can use to generate asynchronous communication between Simulink components and production code.

- If you clear the check box, the block does not accept new messages if the queue is full.

  In this mode, you can manipulate entity data using event actions and visualize statistics.

  To customize actions when entities or messages enter, exit, or are blocked, enter MATLAB code in the `Entry action`, `Exit action`, or `Blocked action` field of the **Event actions** tab. For more information, see "Events and Event Actions".

  For an example, see "Manage Entities Using Event Actions".

Programmatic Use
Block Parameter: QueueOverwriting
Type: character vector
Values: 'on' | 'off'
Default: 'on' (for Simulink) and 'off' (for SimEvents)

Capacity — Specify the capacity of the queue
25 (default) | scalar

Specify the capacity of the queue.

Programmatic Use
Block Parameter: Capacity
Type: character vector
Values: '25' | scalar
Default: '25'

Queue type — Choose the queue type
FIFO (default) | LIFO | Priority

Choose the queue type.

- FIFO — first-in-first-out
- LIFO — last-in-first-out
- Priority — store elements in order of priority, see "Serve High-Priority Customers by Sorting Entities Based on Priority". Priority can be selected when you clear the **Overwrite the oldest element if queue is full** check box.

Note Priority queue does not support `fixed point` data type.

Programmatic Use
Block Parameter: QueueType
Type: character vector
Values: 'FIFO' | 'LIFO' | 'Priority'
Default: 'FIFO'

Multicast tag — Specify the tag when accepting entities broadcast via multicast sources
A (default) | character vector

Specify the tag when accepting entities broadcast via multicast sources. The Entity Multicast block requires SimEvents license.

Dependencies

This parameter is visible when you clear the **Overwrite the oldest element if queue is full** check box, and set **Entity arrival source** to Multicast.

Programmatic Use
Block Parameter: MulticastTag
Type: character vector
Values: 'A' | character vector
Default: 'A'

Priority source — Specify which attribute of the entity determines its priority
PriorityAttribute (default) | character vector

Specify which attribute of the entity determines its priority.

Dependencies

This parameter is visible when you clear the **Overwrite the oldest element if queue is full** check box, and set **Queue type** to Priority.

Programmatic Use
Block Parameter: PrioritySource
Type: character vector
Values: 'PriorityAttribute' | character vector
Default: 'PriorityAttribute'

Sorting direction — Choose the direction of sorting based on priority
Ascending (default) | Descending

Choose the direction of sorting based on priority.

- Ascending — Elements with smaller priority values appear in front of the queue.
- Descending — Elements with greater priority values appear in front of the queue.

Dependencies

This parameter is visible when you clear the **Overwrite the oldest element if queue is full** check box, and set **Queue type** to Priority.

Programmatic Use
Block Parameter: SortingDirection
Type: character vector
Values: 'Ascending' | 'Descending'
Default: 'Ascending'

Entity arrival source — Choose the source of arrival for the entities
Input port (default) | Multicast

Choose the source of arrival for the entities.

- Input port — Input port is source of messages or entities.
- Multicast — Entity Multicast block is source of entities. The Entity Multicast block requires SimEvents license.

Dependencies

This parameter is visible when you clear the **Overwrite the oldest element if queue is full** check box, and set **Queue type** to Priority.

Programmatic Use
Block Parameter: EntityArrivalSource
Type: character vector
Values: 'Input port' | 'Multicast'
Default: 'Input port'

Event action — Specify the behavior of the entity in certain events
Entry (default) | Exit | Blocked

Specify the behavior of the entity in certain events. Define the behavior in the **Event action** parameter. The **Entry** and the **Exit** actions are called just after the entity entry and just before entity exit. The **Blocked** action is called after an entity is blocked. For more information, see "Events and Event Actions".

Note Event actions do not support fixed point data type.

Dependencies

Event actions are visible when you clear the **Overwrite the oldest element if queue is full** check box.

Programmatic Use
Block Parameter: EntryAction, ExitAction, BlockedAction
Type: character vector
Values: MATLAB code
Default: ''

Number of entities departed, d — Outputs the number of entities that have departed the block
off (default) | on

Number of entities that have departed the block.

Dependencies

Number of entities departed, d is visible when you clear the **Overwrite the oldest element if queue is full** check box.

Programmatic Use
Block Parameter: NumberEntitiesDeparted
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities in block, n — Outputs the number of entities present in the block, that are yet to depart
off (default) | on

Number of entities present in the block, but which are yet to depart.

Dependencies

Number of entities in block, n is visible when you clear the **Overwrite the oldest element if queue is full** check box.

Programmatic Use
Block Parameter: NumberEntitiesInBlock
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Average wait, w — Outputs the average wait time
off (default) | on

Sum of the wait times for entities departing the block divided by their total number. Wait time is the duration between the Entity Queue block entry and exit of an entity. For more information, see "Interpret SimEvents Models Using Statistical Analysis".

Dependencies

Average wait, w is visible when you clear the **Overwrite the oldest element if queue is full** check box.

Programmatic Use
Block Parameter: AverageWait
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Average queue length, l — Outputs the average length of the entity queue
off (default) | on

Accumulated time-weighted average queue size. The block computes this value by:

1 Multiplying the size of the queue by its duration to calculate time-weighted queue size
2 Summing up all time-weighted queue sizes and averaging them over total time

For more information, see "Interpret SimEvents Models Using Statistical Analysis".

Dependencies

Average queue length, l is visible when you clear the **Overwrite the oldest element if queue is full** check box.

Programmatic Use
Block Parameter: AverageQueueLength
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities extracted, ex — Number of entities extracted from this block
off (default) | on

Outputs the number of extracted entities which are pulled out from this block by the Entity Find block. The Entity Find block requires a SimEvents license. If the extracted entity is the first entity in the queue, the next entity is set as the pending entity to leave the block. If an entity attribute defines

the priority in a priority queue and it is modified by the Entity Find block, the queue is sorted again. When an entity is extracted, **Number of entities departed, d**, **Number of entities in block, n**, **Average wait, w**, and **Average queue length, l** statistics are updated accordingly. For more information about finding and extracting entities, see "Find and Extract Entities in SimEvents Models".

Dependencies

Number of entities extracted, ex is visible when you clear the **Overwrite the oldest element if queue is full** check box.

Programmatic Use
Block Parameter: NumEntitiesExtracted
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Block Characteristics

Data Types	Boolean | bus | double | enumerated | fixed point | integer | single | string
Direct Feedthrough	no
Multidimensional Signals	yes
Variable-Size Signals	no
Zero-Crossing Detection	no

Extended Capabilities

C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

Code generation is not supported for event actions and statistics.

See Also

Entity Generator | Entity Multicast | Entity Server | Message Receive | Message Send | Multicast Receive Queue

Topics
"Overview of Queues and Servers in Discrete-Event Simulation"
"SimEvents Common Design Patterns"

Introduced in R2016a

Entity Replicator

Replicate entities
Library: SimEvents

Description

The Entity Replicator block duplicates entities. It outputs replica entities and can also output the original entity. The block provides an output port for the original entity.

If the original entity departs the block before the replicas, then its replicas are destroyed. Selecting the **Hold original entity until all replicas depart** check box ensures that the replicas depart the block before the original entity.

An original entity can be extracted from this by block by the Entity Find block. If an original entity waiting in the Entity Replicator block is extracted, all the replicas are destroyed. Only the original entities can be extracted because the replicas can not acquire resources in this block. For more information about finding and extracting entities, see "Find and Extract Entities in SimEvents Models".

Ports

Input

Port_1 — Input entity
scalar | vector | matrix

Input entity port for entities entering the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `string` | `fixed point`

Output

Port_1 — Output replica entity
scalar | vector | matrix

Output entity port for replica entities exiting the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `string` | `fixed point`

Port_2 — Output the entity that is replicated
scalar | vector | matrix

Output entity port for original entities exiting the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `string` | `fixed point`

Parameters

`Replicas depart from` — Choose how replicas depart
`Separate output ports` (default) | `Single output port`

Choose if the replicas depart from separate output ports or a single output port.

- `Separate output ports` — Outputs one replica entity from each output port
- `Single output port` — Outputs all replicas from a single output port

Programmatic Use
Block Parameter: `ReplicasDepartFrom`
Type: character vector
Values: `'Separate output ports'` | `'Single output port'`
Default: `'Separate output ports'`

`Replication amount source` — Specify source of replica number
`Dialog` (default) | `Attribute`

Specify the source of replica number.

- Select `Dialog` to specify the number of replicas in the dialog box.
- Select `Attribute` to select an attribute that specifies the number of replicas.

Dependencies

This parameter is visible when you set **Replicas depart from** to `Single output port`.

Programmatic Use
Block Parameter: `ReplicationAmountSource`
Type: character vector
Values: `'Dialog'` | `'Attribute'`
Default: `'Dialog'`

`Number of replicas` — Specify the number of replicas
1 (default) | scalar

Specify the number of replicas. 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.

Dependencies

This parameter is visible when you set **Replicas depart from** to `Separate output ports` or `Single output port` and **Replication amount source** to `Dialog`.

Programmatic Use
Block Parameter: `NumberReplicas`
Type: character vector
Values: `'1'` | scalar
Default: `'1'`

`Replicate attribute name` — Specify the attribute that determines the number of replicas
`ReplicateAttribute` (default) | character vector

Dependencies

Specify the attribute that determines the number of replicas. This parameter is visible when you set **Replicas depart from** to `Single output port` and **Replication amount source** to `Attribute`.

Programmatic Use
Block Parameter: ReplicateAttributeName
Type: character vector
Values: 'ReplicateAttribute' | character vector
Default: 'ReplicateAttribute'

Hold original entity until all replicas depart — Choose how replicas depart
off (default) | on

Select this check box to hold the original entity until all the replicas have departed. The block first attempts to send all the replicas before it sends out the original entity. Selecting this check box prevents destruction of replica entities when they do not depart the block before the original entity.

Programmatic Use
Block Parameter: HoldOriginalEntityUntilAllReplicasDepart
Type: character vector
Values: 'on' | 'off'
Default: 'off'

See Also
Entity Generator | Entity Server

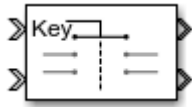**Topics**
"Enable a Gate for a Time Interval"
"SimEvents Common Design Patterns"

Introduced in R2016a

Entity Selector

Select entities
Library: SimEvents

Description

The Entity Selector block selects entities from multiple streams of ready-to-leave entities, and matches them to a reference entity. You can use this block for modeling scenarios which involve items to be matched based on input. For instance, you can model a facility that produces parts which are matched to the corresponding order.

The block first accepts a reference entity with its **Key entity attribute name**. Then the block selects a matching entity with **Matching entity attribute name(s)** from each of the other input ports accepting incoming entities. The match is based on the equality of the specified attribute values. When a match is found across all the entity streams, the set of matching entities and the key entity become ready to depart.

The Entity Store block can be used as a temporary container for entities to be selected by the Entity Selector block. For an example, see "Match Entities Based on Attributes".

Ports

Input

Key — Incoming reference entity
scalar | vector | matrix

Input entity port for reference entities entering the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_1 — Incoming entity
scalar | vector | matrix

Input entity port for matching entities entering the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Output

Port_1 — Exiting reference entity
scalar | vector | matrix

Output entity port for reference entities exiting the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_2 — Exiting matched entity
scalar | vector | matrix

Output entity port for matching entities exiting the selector.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_d — Number of entities that have departed the block
scalar

Number of entities that have departed the block.

Dependencies

To enable this port, click **Statistics** and select the **Number of entities departed, d** check box.

Data Types: `double`

Port_n — Number of entities that have not yet departed the block
scalar

Number of entities that have not yet departed the block.

Dependencies

To enable this block, click **Statistics** and select the **Number of entities in block, n** check box.

Data Types: `double`

Port_ex — Number of entities extracted
scalar

Number of entities that are pulled out of this block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities extracted, ex**.

Data Types: `double`

Parameters

`Number of matching streams` — Determine how many input entity streams the block accepts
1 (default) | numerical

Specify the number of entity streams to be matched to the reference entity. The block can accept at most 8 matching streams.

Programmatic Use
Block Parameter: `mNumberOfStreams`
Type: character vector

Values: '1' | scalar
Default: '1'

Key entity attribute Name — Specify name of reference attribute
{'ID'} (default) | character vector

Name of the reference attribute that is used to evaluate a match.

Programmatic Use
Block Parameter: mKeyEntityAttributeName
Type: character vector
Values: '{'ID'}' | character vector
Default: '{'ID'}'

Matching entity attribute name(s) — Specify name of selected attribute names
{'Type'} (default) | character vector

Name of the matching entity attribute that is used to evaluate a match. You can specify one attribute name, or an array of attribute names that are compared with the key value to evaluate a match.

Programmatic Use
Block Parameter: mEntityAttributeName
Type: character vector
Values: '{'Type'}' | character vector
Default: '{'Type'}'

Number of entities source — Source to determine the number of entities to be matched
Dialog (default) | Attribute

Source that determines the number of entities to be selected from each stream.

Programmatic Use
Block Parameter: mMatchingNumberSource
Type: character vector
Values: 'Dialog' | 'Attribute'
Default: 'Dialog'

Number of entities to select — Number of entities to select from each of the incoming entity streams
1 (default) | scalar

Number of entities to be selected from each matching stream. You can specify 0, a positive integer, or an array of positive integers to determine the number of entities to select from each matching entity stream.

Dependencies

To enable this parameter, set the **Number of entities source** parameter to Dialog.

Programmatic Use
Block Parameter: mNumberOfMatches
Type: character vector
Values: '1' | scalar
Default: '1'

Key attribute for number of entities — Key attribute name that determines the number of entities to be matched
Name (default) | character vector

The name of the key attribute that determines the number of entities to be selected from each matching stream. The attribute value can be an integer or an array of integers of size equal to the number of incoming entity streams.

Dependencies

To enable this parameter, set the **Number of entities source** parameter to Attribute.

Programmatic Use
Block Parameter: mNumberOfMatchesAttribute
Type: character vector
Values: 'Name' | character vector
Default: 'Name'

Number of entities departed, d — Outputs the number of entities that have departed the block
off (default) | on

Number of entities that have exited the block.

Programmatic Use
Block Parameter: mNumEntitiesDepOpt
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities in block, n — Outputs the number of entities present in the block, which have yet to depart
off (default) | on

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

Programmatic Use
Block Parameter: mNumEntitiesInBlockOpt
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities extracted, ex — Outputs the number of entities extracted from the block
off (default) | on

Outputs the number of extracted entities which are pulled out from this block by the Entity Find block. If a key entity is extracted, the Entity Selector block waits until all the matching entities arrive at the block. Then, the matching entities depart from the corresponding output port. If a matching entity is extracted, the block looks for another matching entity. When an entity is extracted, **Number of entities departed, d**, and **Number of entities in block, n** statistics are updated accordingly.

Programmatic Use
Block Parameter: mNumEntitiesExtractedOpt
Type: character vector
Values: 'on' | 'off'

Default: `'off'`

See Also

Composite Entity Creator | Composite Entity Splitter | Discrete Event Chart | Entity Store | Entity Input Switch | Entity Multicast | Entity Output Switch | Entity Queue | Entity Replicator | Entity Server
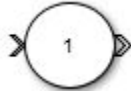
Topics
"SimEvents Common Design Patterns"
"Match Entities Based on Attributes"

Introduced in R2018a

Entity Server

Serve entities
Library: SimEvents

Description

The Entity Server block serves entities as they arrive. In a discrete-event simulation, a server stores entities for a length of time, called service time, and then attempts to output the entity. During the service period, the block is said to be serving the entity that it stores. The block 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.

To customize actions when entities enter, complete service, exit, and are blocked or preempted by the block, enter MATLAB code in the `Entry action`, `Service complete action`, `Exit action`, `Blocked action`, or `Preempt action` field of the **Event actions** tab. For more information, see "Events and Event Actions".

Ports

Input

Port_1 — Input entity
scalar | vector | matrix

Input entity that carries scalar, bus, or vector data to enter the server.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `string` | `fixed point`

Port_2 — Input signal port for service time source
scalar

Input signal port to determine service time.

Dependencies

This port is visible when **Service time source** is set to `Signal port`.

Data Types: `double`

Output

Port_1 — Output entity
scalar | vector | matrix

Output entity port for entities exiting the server.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `string` | `fixed point`

Port_d — Number of entities that have departed the block
scalar

Number of entities that have departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities departed, d**.

Data Types: `double`

Port_n — Number of entities that have not yet departed the block
scalar

Number of entities that have not yet departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities in block, n**.

Data Types: `double`

Port_pe — Pending entity in block
off (default) | on

Selecting this check box outputs the value 1 for a pending entity in the block, and 0 otherwise.

Dependencies

To enable this port, select the **Statistics > Pending entity in block, pe**.

Data Types: `double`

Port_np — Number of pending entities
off (default) | on

Selecting this check box outputs the number of pending entities in the block.

Dependencies

To enable this port, select the **Statistics > Number of pending entities, np**.

Data Types: `double`

Port_w — Average wait time for entities in the block
scalar

Average wait time for entities in the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Average wait, w**.

Data Types: `double`

Port_util — Outputs the average time the server is occupied
scalar

Average time the server is occupied.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Utilization, util**.

Data Types: `double`

Port_p — Outputs the number of preempted entities
scalar

Number of preempted entities.

Dependencies

To enable this port, first click the **Preemption** tab, then click **Statistics** tab and select the box labeled **Number of entities preempted, p**.

Data Types: `double`

Port_ex — Number of entities extracted
scalar

Number of entities that are pulled out of this block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities extracted, ex**.

Data Types: `double`

Parameters

Capacity — Specify the capacity of the server
1 (default) | scalar

Specify the number of entities the block can serve simultaneously.

Programmatic Use
Block Parameter: `Capacity`
Type: character vector
Values: `'1'` | scalar
Default: `'1'`

Service time source — Choose the source to specify the service time
Dialog (default) | Signal port | Attribute | MATLAB action

Determine the source that specifies the service time.

You can select:

* `Dialog`

  Enter the constant value in the **Service time value** parameter.
* `Signal port`

  Connect a time source to the resulting signal port.

- Attribute

    Enter the name of the attribute that contains data to be interpreted as service.

- MATLAB action

    In the **Service time action** section, enter MATLAB code to vary the service time. Assign the variable *dt*, which the model uses as service time.

Programmatic Use
Block Parameter: ServiceTimeSource
Type: character vector
Values: 'Dialog' | 'Signal port' | 'Attribute' | 'MATLAB action'
Default: 'Dialog'

Service time attribute name — Specify service time source attribute name
ServiceTime (default) | character vector

Determine the name of the attribute that is used as the service time value.

Dependencies

This parameter is visible when **Service time source** is set to Attribute.

Programmatic Use
Block Parameter: ServiceTimeAttributeName
Type: character vector
Values: 'ServiceTime' | character vector
Default: 'ServiceTime'

Service time value — Specify the value of the service time
1 (default) | scalar

Tunable: Yes

Dependencies

This parameter is visible when **Service time source** is set to Dialog.

Programmatic Use
Block Parameter: ServiceTimeValue
Type: character vector
Values: '1.0' | scalar
Default: '1.0'

Service time action — Specify service time
dt = rand(1,1) (default) | MATLAB code

Use MATLAB code to specify service time. dt specifies the service time. You can manually specify dt or use **Insert pattern** button to specify service time from a repeating sequence or from a distribution. For an example, see "Count Simultaneous Departures from a Server".

Dependencies

This parameter is visible when **Service time source** is set to MATLAB action.

Programmatic Use
Block Parameter: ServiceTimeAction

Type: character vector
Values: MATLAB code
Default: `'dt = rand(1,1);'`

Event action — Specify the behavior of the entity in certain events
Entry (default) | Service complete | Exit | Blocked | Preempt

Specify the behavior of the entity in certain events. Define the behavior in the **Event action** parameter. The **Entry** and the **Exit** actions are called just after the entity entry and just before entity exit. The **Service complete** action is called after the completion of the entity service. The **Blocked** action is called after an entity is blocked. The **Preempt** is called after the preemption.

Programmatic Use
Block Parameter: EntryAction, ServiceCompleteAction, ExitAction, BlockedAction, PreemptAction
Type: character vector
Values: MATLAB code
Default: `''`

Permit preemption based on attribute — Enable preemption of entities
off (default) | on

Select this check box if you want to allow preemption of entities. Preemption is the replacement of an entity in a server block by an entity that satisfies certain criteria. Selecting this check box enables these parameters:

- **Sorting attribute name** in the **Preemption** tab
- **Sorting direction** in the **Preemption** tab
- **Write residual time to attribute** in the **Preemption** tab
- **Number of entities preempted, p** in the **Statistics** tab

For an example, see "Task Preemption in a Multitasking Processor".

Programmatic Use
Block Parameter: PermitPreemptionBasedOnAttribute
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Sorting attribute name — Specify the name of the attribute used to determine the priority
entity | entitySys.priority | character vector

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

Dependencies

This parameter is visible when the **Permit preemption based on attribute** box is selected.

Programmatic Use
Block Parameter: SortingAttributeName
Type: character vector
Values: `'entity'` | `'entitySys.priority'` | character vector
Default: `'entity'`

`Sorting direction` — Choose the direction of sorting the entities
Ascending (default) | Descending

Specify if the entities are sorted in `ascending` or `descending` order.

- `ascending` — Sorting entities with smaller key values to have a higher priority
- `descending` — Sorting entities with greater key values to have a higher priority

Dependencies

This parameter is visible when the **Permit preemption based on attribute** box is selected.

Programmatic Use
Block Parameter: SortingDirection
Type: character vector
Values: 'Ascending' | 'Descending'
Default: 'Ascending'

`Write residual time to attribute` — Save the residual service time from a preempted entity to an attribute
off (default) | on

Dependencies

This parameter is visible when the **Permit preemption based on attribute** box is selected.

Programmatic Use
Block Parameter: WriteResidualTimeToAttribute
Type: character vector
Values: 'on' | 'off'
Default: 'off'

`Residual time attribute name` — Specify the name of the attribute to contain the residual service time of a preempted entity
ResidualTime (default) | character vector

Dependencies

This parameter is visible when the **Write residual time to attribute** box is selected.

Programmatic Use
Block Parameter: ResidualTimeAttributeName
Type: character vector
Values: 'ResidualTime' | character vector
Default: 'ResidualTime'

`Number of entities departed, d` — Outputs the number of entities that have departed the block
off (default) | on

Number of entities that have departed the block.

Programmatic Use
Block Parameter: NumberEntitiesDeparted
Type: character vector
Values: 'on' | 'off'

Default: `'off'`

Number of entities in block, n — Outputs the number of entities present in the block, that are yet to depart
off (default) | on

Number of entities present in the block that are yet to depart.

Programmatic Use
Block Parameter: `NumberEntitiesInBlock`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Pending entity in block, pe — Pending entities
off (default) | on

Indicates whether an entity that is yet to depart is present in the block. The value is 1 for a pending entity, and 0 otherwise.

Programmatic Use
Block Parameter: `PendingEntityPresentInBlock`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Number of pending entities, np — Number of Pending entities
off (default) | on

Outputs the number of pending entities the block has served that are yet to depart.

Programmatic Use
Block Parameter: `NumberEntitiesPending`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Average wait, w — Outputs the average wait time
off (default) | on

Sum of the wait times for entities departed the block divided by their total number. Wait time is the duration between the Entity Server block entry and exit of an entity. For more information, see"Interpret SimEvents Models Using Statistical Analysis".

Programmatic Use
Block Parameter: `AverageWait`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Utilization, util — Outputs the average time the server is occupied
off (default) | on

Average time the server is occupied. The block calculates this time as the ratio of the total wait time for entities to the server capacity multiplied by the total simulation time.

Wait time is the duration between the Entity Server block entry and exit of an entity. Total wait time is the sum of the wait times for entities departed the block.

Programmatic Use
Block Parameter: `Utilization`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Number of entities preempted, p — Outputs the number of preempted entities
off (default) | on

Outputs the number of preempted entities. Preemption is the replacement of an entity in a server block by an entity that satisfies certain criteria.

Dependencies

This check box appears if the **Permit preemption based on attribute** check box is selected.

Programmatic Use
Block Parameter: `NumberEntitiesPreempted`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Number of entities extracted, ex — Number of entities extracted from this block
off (default) | on

Outputs the number of extracted entities which are pulled out from this block by the Entity Find block. If an entity is being served during the extraction, the service is terminated. If an attribute that defines the service time and it is modified by the Entity Find block, service time does not change. When an entity is extracted, **Number of entities departed, d**, **Number of entities in block, n**, **Average wait, w**, and **Utilization, util** statistics are updated accordingly. For more information about finding and extracting entities, see "Find and Extract Entities in SimEvents Models".

Programmatic Use
Block Parameter: `NumEntitiesExtracted`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

See Also

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

Topics
"Count Simultaneous Departures from a Server"
"Model Server Failure"
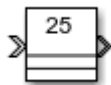"Overview of Queues and Servers in Discrete-Event Simulation"
"Storage with Servers"
"SimEvents Common Design Patterns"

Introduced in R2016a

Entity Store

Store entities
Library: SimEvents

Description

The Entity Store block serves as a container or bin to store unordered entities. The entities are ready to leave the block immediately provided they are accepted by the next block. The Entity Store block attempts to forward an entity immediately upon its arrival. If the attempt fails, the block puts the entity in a pending state. The entity can then leave when the next block can start accepting it.

To customize actions when entities enter, exit, or are blocked, enter MATLAB code in the `Entry action`, `Exit action`, or `Blocked action` field of the **Event actions** tab.

Ports

Input

Port_1 — Input entity
scalar | vector | matrix

Input entity port for entities entering the storage.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Output

Port_1 — Output entity
scalar | vector | matrix

Output entity port for entities exiting the storage.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_d — Number of entities that have departed the block
scalar

Number of entities that have departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities departed, d**.

Data Types: `double`

Port_n — Number of entities that have not yet departed the block
scalar

Number of entities that have not yet departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities in block, n**.

Data Types: `double`

Port_w — Average wait time for entities in the block
scalar

Average wait time for entities in the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Average wait, w**.

Data Types: `double`

Port_l — Average store size
scalar

Average size of the store.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Average store size, l**.

Data Types: `double`

Port_ex — Number of entities extracted
scalar

Number of entities that are pulled out of this block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities extracted, ex**.

Data Types: `double`

Parameters

Capacity — Specify the capacity of the storage
25 (default) | scalar

Specify the maximum number of entities contained in the storage.

Programmatic Use
Block Parameter: Capacity
Type: character vector
Values: '25' | scalar
Default: '25'

Number of entities departed, d — Outputs the number of entities that have departed the block
off (default) | on

Selecting this check box outputs the number of entities that have exited the block.

Programmatic Use
Block Parameter: NumberEntitiesDeparted
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities in block, n — Outputs the number of entities present in the block, which have yet to depart
off (default) | on

Selecting this check box outputs the number of entities present in the block, which have yet to depart.

Programmatic Use
Block Parameter: NumberEntitiesInBlock
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Average wait, w — Outputs the average wait time
off (default) | on

Sum of the wait times for entities departing the block divided by their total number. Wait time is the duration between the Entity Store block entry and exit of an entity. For more information, see "Interpret SimEvents Models Using Statistical Analysis".

Programmatic Use
Block Parameter: AverageWait
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Average store size, l — Outputs the average store size
off (default) | on

Accumulated time-weighted average store size. Store size is the number of entities stored in the block. The block computes average store size by:

1 Multiplying the store size by its duration to calculate time-weighted store size

2 Summing up all time-weighted store sizes and averaging them over total time

For more information, see "Interpret SimEvents Models Using Statistical Analysis".

Programmatic Use
Block Parameter: AverageStoreSize
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities extracted, ex — Number of entities extracted from this block
off (default) | on

Outputs the number of extracted entities which are pulled out from this block by the Entity Find block. When an entity is extracted, **Number of entities departed, d**, **Number of entities in block, n**, **Average wait, w**, and **Average store size, l** statistics are updated accordingly. For more information about finding and extracting entities, see "Find and Extract Entities in SimEvents Models".

Programmatic Use
Block Parameter: NumEntitiesExtracted
Type: character vector
Values: 'on' | 'off'
Default: 'off'

See Also

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

Topics
"SimEvents Common Design Patterns"
"Match Entities Based on Attributes"

Introduced in R2018a

Entity Terminator

Terminate entities
Library: SimEvents

Description

The Entity Terminator block accepts and destroys entities. Use this block to represent the entity departure from the model.

To customize actions when entities enter, use MATLAB code in the `Entry action` field of the **Event actions** tab. See "Events and Event Actions", for more information.

Ports

Input

Port_1 — Input entity
scalar | vector | matrix

Input entity port for entities entering the terminator.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Output

Port_a — Number of entities arrived
scalar

Number of entities that have arrived at the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities arrived, a**.

Data Types: `double`

Parameters

Event action — Specify the behavior of the entity in certain events
Entry

Specify the behavior of the entity in certain events. Define the behavior in the **Event action** parameter. The **Entry** action is called just after the entity entry to the block.

Programmatic Use
Block Parameter: EntryAction

Type: character vector
Values: MATLAB code
Default: `''`

Number of entities arrived, a — Outputs the number of entities that have arrived at the block
off (default) | on

Number of entities that have arrived at the block.

Programmatic Use
Block Parameter: `NumberEntitiesArrived`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

See Also

Entity Generator | Entity Queue | Entity Server

Topics
"SimEvents Common Design Patterns"

Introduced in R2016a

Hit Crossing

Detect crossing point

Library: Simulink / Discontinuities
                    Simulink / Messages & Events
                    HDL Coder / Discontinuities
                    SimEvents

Description

The Hit Crossing block detects when the input reaches the **Hit crossing offset** parameter value in the direction specified by the **Hit crossing direction** property.

You can configure the block to output a `1` or `0` signal, a SimEvents message, or a function-call event. See "Output" on page 2-81 for more information.

Ports

Input

Port_1 — Input signal
scalar | vector

Input signal that the block detects when it reaches the offset in the specified direction.

Data Types: `double`

Output

Port_1 — Output signal
scalar | vector | SimEvents message | function-call event

Output indicating if the input signal crossed the hit offset. This port is visible only when you select the **Show output port** parameter check box.

Signal Output

If you select the **Show output port** check box and set the **Output type** parameter to `Signal`, the block output indicates when the crossing occurs.

- If the input signal is exactly the value of the offset value after the hit crossing is detected in the specified direction, the block continues to output a value of 1.
- If the input signals at two adjacent points brackets the offset value, the block outputs a value of 1 at the second time step.
- If the **Show output port** check box is *not* selected, the block ensures that the simulation finds the crossing point but does not generate output.
- If the initial signal is equal to the offset value, the block outputs 1 only if the **Hit crossing direction** property is set to `either`.

- If Boolean logic signals are enabled, then the output is a `Boolean`.

SimEvents Message Output

The Hit Crossing block can also output a SimEvents message when the **Output type** parameter is set to `Message`.

- If the input signal crosses the offset value in the specified direction, the block outputs a message.
- If the input signal reaches the offset value in the specified direction and remains there, block outputs one message at the hit time and one message when the signal leaves the offset value.
- If the initial input signal is equal to the offset value, the block outputs a message with `Crossing Type` value `None` only if the **Hit crossing direction** is set to `either`.

The SimEvents message output signal is a `struct` with four fields.

Function-Call Output

The Hit Crossing block can also output a function-call event when the **Output type** parameter is set to `Function-Call`.

- Each time the input signal crosses the offset value in the specified direction, the block outputs a single function-call event.
- The function-call event can be sent to the function-call input port of a function-call subsystem or function-call model.
- The output is equivalent to the output of a Function-Call Generator block at each time step with the **Number of iterations** parameter of that block set to `1`.

CrossingType — Direction of zero-crossing
None | NegativeToPositive | NegativeToZero | ZeroToPositive | PositiveToNegative | PositiveToZero | ZeroToNegative

This field shows the direction in which the signal crosses the **Hit crossing offset** value. Negative, Zero, and Positive are defined relative to the offset value. The data type is `slHitCrossingType` which is an enumerated data type. See "Use Enumerated Data in Simulink Models" for more information. For example, if `HitCrossingOffset` is set to 2, a rising signal crossing this offset value would be recorded as a `NegativeToPositive` hit crossing.

Note A hit crossing is recorded based on the **Hit crossing direction** setting. In other words, if you set **Hit crossing direction** to detect a `falling` hit crossing, a `NegativeToPositive` hit is not recorded.

Note In a SimEvents block, if the `Crossing Type` of an entity is a `NegativeToPositive` hitcrossing then `entity.CrossingType == slHitCrossingType.NegativeToPositive` returns logical 1 (`true`).

If the signal reaches the `HitCrossingOffset` value and holds it, a single `NegativeToZero` or `PositiveToZero`, depending on the direction, hit is registered at the time of the hit crossing.

Data Types: `slHitCrossingType`

Index — Index of the input signal at which the hit crossing event occurs
nonnegative integer

For *n* signals being passed to the Hit Crossing block, this field denotes which signal had a hit crossing event. For a matrix input, this field follows MATLAB linear indexing. See "Array Indexing".

Data Types: `uint32`

`Time` — Time of hit crossing event
real, finite

Time *T* of the hit crossing event.

Data Types: `double`

`Offset` — Hit crossing value for detection
0 (default) | real values

Hit crossing offset value as specified by the "Hit crossing offset" parameter.

Data Types: `double`

Data Types: `double` | `Boolean` | `struct`

Note If the SimEvents message output signal crosses model reference boundaries or is used as an input to a Stateflow chart, you need to create a bus object for the message. See "Tips".

Parameters

`Hit crossing offset` — Hit crossing value for detection
0 (default) | real values

Specify the value the block detects when the input crosses in the direction specified by **Hit crossing direction**.

Programmatic Use
Block Parameter: `HitCrossingOffset`
Type: character vector
Values: real values
Default: `'0'`

`Hit crossing direction` — Input signal direction to hit crossing
either (default) | falling | rising

Direction from which the input signal approaches the hit crossing offset for a crossing to be detected.

When set to `either`, the block serves as an *almost equal* block, useful in working around limitations in finite mathematics and computer precision. Used for these reasons, this block might be more convenient than adding logic to your model to detect this condition.

When the **Hit crossing direction** property is set to `either` and the model uses a fixed-step solver, the block has the following behavior. If the output signal is 1, the block sets the output signal to 0 at the next time step, unless the input signal equals the offset value.

Programmatic Use
Block Parameter: `HitCrossingDirection`
Type: character vector

Values: 'either' | 'rising' | 'falling'
Default: 'either'

Show output port — Display an output port
off (default) | on

If selected, create an output port on the block icon.

Programmatic Use
Block Parameter: ShowOutputPort
Type: character vector
Values: 'off' | 'on'
Default: 'on'

Output type — Choose signal, message, or function-call output
Signal (default for Simulink) | Message (default for SimEvents) | Function-Call

When **Output type** is set to Signal, the output signal is set to one whenever the input signal crosses the **Hit crossing offset** value in the **Hit crossing direction** and is zero at other times.

When the **Output type** is set to Message, the output signal becomes a SimEvents message.

When **Output type** is set to Function-Call, the output signal becomes a function-call event.

Programmatic Use
Block Parameter: HitCrossingOutputType
Type: character vector
Values: 'Signal' | 'Message' | 'Function-Call'
Default: 'Signal'

Enable zero-crossing detection — Enable zero-crossing detection
on (default) | off

Select to enable zero-crossing detection. For more information, see "Zero-Crossing Detection".

Programmatic Use
Parameter: ZeroCross
Type: character vector, string
Values: 'on' | 'off'
Default: 'on'

Block Characteristics

Data Types	double
Direct Feedthrough	yes
Multidimensional Signals	no
Variable-Size Signals	no
Zero-Crossing Detection	yes

Tips

If the Hit Crossing block is configured to output a SimEvents message and the output signal:

- Crosses into or out of a referenced model
- Is fed to the input of a Stateflow chart

then you need to create a bus object for the message signal. In the MATLAB Command Window, run `Simulink.createHitCrossMessage` to check for and, if needed, create a hit crossing message bus object in the base workspace.

Set the data type of the corresponding port to `Bus: HitCrossMessage`.

Extended Capabilities

C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

Not recommended for production code.

Does not support non-floating data type for ert targets.

HDL Code Generation
Generate Verilog and VHDL code for FPGA and ASIC designs using HDL Coder™.

HDL Coder™ provides additional configuration options that affect HDL implementation and synthesized logic.

HDL Architecture

This block has a single, default HDL architecture.

HDL Block Properties

ConstrainedOutputPipeline	Number of registers to place at the outputs by moving existing delays within your design. Distributed pipelining does not redistribute these registers. The default is 0. For more details, see "ConstrainedOutputPipeline" (HDL Coder).
InputPipeline	Number of input pipeline stages to insert in the generated code. Distributed pipelining and constrained output pipelining can move these registers. The default is 0. For more details, see "InputPipeline" (HDL Coder).
OutputPipeline	Number of output pipeline stages to insert in the generated code. Distributed pipelining and constrained output pipelining can move these registers. The default is 0. For more details, see "OutputPipeline" (HDL Coder).

Restriction

The Hit crossing direction must be `rising` or `falling`.

HDL code generation is not supported when the `Output Type` is set to `Message`.

See Also

"Zero-Crossing Detection" | "Implement logic signals as Boolean data (vs. double)"

Introduced in R2018a

MATLAB Discrete-Event System

MATLAB discrete-event system
Library: SimEvents

Description

The MATLAB Discrete-Event System block allows you to create and author custom discrete-event systems. With this block, you can author an event-driven entity-flow system using MATLAB, and use it in your Simulink model. For more information about implementing `matlab.DiscreteEventSystem` class with MATLAB Discrete-Event System block, see "Create Custom Blocks Using MATLAB Discrete-Event System Block".

Below, there are six examples to help you learn how to create custom blocks using the MATLAB Discrete-Event System block.

1   "Delay Entities with a Custom Entity Storage Block"
2   "Create a Custom Entity Storage Block with Iteration Event"
3   "Custom Entity Storage Block with Multiple Timer Events"
4   "Custom Entity Generator Block with Signal Input and Signal Output"
5   "Build a Custom Block with Multiple Storages"
6   "Create a Custom Resource Acquirer Block"

Parameters

`System object name` — Specify the full name of the System object
character vector

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, the name appears 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.

Programmatic Use
Block Parameter: System
Type: character vector
Values: '<Enter System Class Name>' | character vector
Default: '<Enter System Class Name>'

New — Create a SimEvents System object from a template
SimEvents System object

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.

Simulate using — Specify the simulation mode
Code generation (default) | Interpreted execution

Specify the simulation mode as Code generation or Interpreted execution.

- Interpreted execution

  This mode simulates the System object based on the interpreted MATLAB language with debuggers enabled.

- Code generation

  This code generation mode reduces simulation time of SimEvents models. On the first model run, the MATLAB Discrete-Event System block simulates and generates code using only MATLAB functions supported for code generation. If the System object code and the block parameters do not change, subsequent model runs do not regenerate the code. MATLAB Discrete-Event System blocks also support code reuse for models that have multiple MATLAB Discrete-Event System blocks using the same System object source file. For more information, see Generate Code for MATLAB Discrete-Event System Blocks.

Programmatic Use
Block Parameter: SimulateUsing
Type: character vector
Values: 'Code generation' | Interpreted execution
Default: 'Code generation'

See Also

Discrete Event Chart | matlab.DiscreteEventSystem | matlab.System

Topics
"Create Custom Blocks Using MATLAB Discrete-Event System Block"
"Call Simulink Function from a MATLAB Discrete-Event System Block"
"Generate Code for MATLAB Discrete-Event System Blocks"
"SimEvents Common Design Patterns"

Introduced in R2016a

Receive, Message Receive

Receive messages

Library: Simulink / Messages & Events
                     SimEvents

Description

The 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. Receive and Message Receive blocks are identical blocks.

- Select `Hold last value` to hold data read from the last message.

- Select `Use initial value` to write default data.

Ports

Input

Port_1 — Input message
scalar | vector | matrix

The input port for the message.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Output

Port_S — Show whether a message was received
scalar

Outputs `1` if the block receives a message successfully, and `0` otherwise.

Dependencies

To enable this port, select the check box labeled **Show receive status**.

Data Types: `double`

Port_1 — Output signal
scalar | vector | matrix

Output port for the signal.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Parameters

Use internal queue — Select to use an internal queue
on (default for SimEvents) | off (default for Simulink)

Select this check box if you use an internal queue to receive messages.

Programmatic Use
Block Parameter: UseInternalQueue
Type: character vector
Values: 'on' | 'off'
Default: 'on'

Overwrite the oldest element if queue is full — Specify queue overwriting policy
on (default for Simulink) | off (default for SimEvents)

Select this check box to choose between two queue overwriting policies.

- If you select the check box, an incoming message overwrites the oldest if the queue is full.
- If you clear the check box, the block does not accept new messages if the queue is full.

Dependencies

This parameter is visible when you select the box labeled **Use internal queue**.

Programmatic Use
Block Parameter: QueueOverwriting
Type: character vector
Values: 'on' | 'off'
Default: 'on' (for Simulink) and 'off' (for SimEvents)

Queue length — Specify the length of the message queue
16 (default) | scalar

Specify message queue capacity. The queue length can be specified as a positive integer between 1 and 2^{26}-1 (both included).

Dependencies

This parameter is visible when you select the box labeled **Use internal queue**.

Programmatic Use
Block Parameter: QueueLength
Type: character vector
Values: '16' | scalar
Default: '16'

Queue type — Specify message queue sorting policy
FIFO (default) | LIFO | Priority

The block supports three message sorting policies:

- First-in-first-out (FIFO) — The oldest message in the storage departs first.
- Last-in-first-out (LIFO) — The newest message in the storage departs first.
- Priority — Messages or entities are sorted based on their priority.

The priority queue can be used only when the **Overwrite the oldest element if queue is full** check box is cleared.

> **Note** Priority queue accepts only non-bus scalar and it does not support `fixed point` data type.

Dependencies

This parameter is visible when you select the box labeled **Use internal queue**.

Programmatic Use
Block Parameter: QueueType
Type: character vector
Values: 'FIFO' | 'LIFO' | 'Priority'
Default: 'FIFO'

Priority order — Specify message queue priority
Ascending (default) | Descending

Choose the direction of sorting messages based on priority.

- Ascending — Messages with smaller priority values appear in front of the queue.
- Descending — Messages with greater priority values appear in front of the queue.

Dependencies

This parameter is visible when you select the box labeled **Use internal queue** and**Queue type > Priority**.

Programmatic Use
Block Parameter: PriorityOrder
Type: character vector
Values: 'Ascending' | 'Descending'
Default: 'Ascending'

Show receive status — Show whether a message was received
off (default) | on

Select this check box to show whether a message was received. If this check box is selected the block outputs 1 if it receives a message successfully, and 0 otherwise.

Programmatic Use
Block Parameter: ShowQueueStatus
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Initial value — Set initial data value
[] (unspecified) (default) | scalar | vector | matrix

Enter an initial data value for the queue before the arrival of the first message. The default value [] (unspecified) is treated as 0 with data type double.

To use this block to initialize a nonvirtual bus signal, specify the initial value as a MATLAB structure. For more information about initializing nonvirtual bus signals using structures, see "Specify Initial Conditions for Bus Elements".

Programmatic Use
Block Parameter: InitialValue
Type: character vector
Values: '[]' | scalar
Default: '[]'

Value source when queue is empty — Value source for empty queue
Hold last value (default) | Use initial value

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

*   Hold last value (default) — Holds data read from the last message.

    Initially, if the block tries to receive a message and fails, it outputs the initial value.
*   Use initial value — Writes default data.

Dependencies

This parameter is visible when you select the box labeled **Use internal queue**.

Programmatic Use
Block Parameter: ValueSourceWhenQueueIsEmpty
Type: character vector
Values: 'Hold last value' | 'Use initial value'
Default: 'Hold last value'

Sample time (-1 for inherited) — Specify the time interval between samples
-1 (default) | scalar

To inherit the sample time, set this parameter to -1. See "Specify Sample Time" for more information.

Programmatic Use
Block Parameter: SampleTime
Type: character vector
Values: '-1' | scalar
Default: '-1'

Block Characteristics

Data Types	Boolean	bus	double	enumerated	fixed point	integer	single	string
Direct Feedthrough	no
Multidimensional Signals	yes
Variable-Size Signals	no
Zero-Crossing Detection	no

Extended Capabilities

C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

See Also

Message Send

Topics
"Simulink Messages Overview"
"SimEvents Common Design Patterns"

Introduced in R2016a

Send, Message Send

Create and send message
Library:　　　　Simulink / Messages & Events
　　　　　　　　　SimEvents

Description

The Send block reads the value of an input signal, and sends a message that carries this value. In message-based communication, a message is a discrete-item of interest that carry data of any type that Simulink supports. Send and Message Send blocks are identical blocks.

Ports

Input

Port_Enable — External enable signal
scalar

Input port to enable the block to send a message. For any input value that is greater than `0` send is enabled. For any value less than or equal to `0`, the send is disabled.

Dependencies

To enable this port, select the box labeled **Show enable port**.

Data Types: `double`

Port_1 — Input signal
scalar | vector | matrix

This block accepts inputs of any type that Simulink supports, including enumerated types and converts the input signal to a message. For more information, see "Data Types Supported by Simulink".

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Output

Port_1 — Output message
scalar | vector | matrix

The block outputs a message with constant priority `20`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Parameters

Show enable port — Display the enable port
off (default) | on

Select this check box to display enable port. For any input value that is greater than 0 send is enabled. For any value less than or equal to 0, the send is disabled.

Programmatic Use
Block Parameter: ShowEnablePort
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Block Characteristics

Data Types	Boolean | bus | double | enumerated | fixed point | integer | single | string
Direct Feedthrough	no
Multidimensional Signals	yes
Variable-Size Signals	no
Zero-Crossing Detection	no

Extended Capabilities

C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

See Also

Message Receive

Topics
"Simulink Messages Overview"
"SimEvents Common Design Patterns"
"Generate Entities When Events Occur"
"Enable a Gate for a Time Interval"

Introduced in R2016a

Entity Transport Delay

Introduce a delay in propagation of a SimEvents message
Library: Simulink / Continuous
                      SimEvents

Description

The Entity Transport Delay block delays an entity for a period of time, named *transport delay*. The first input is the entity that is transported from point A to point B on a constant-length moving surface whose speed changes over time. The value from the second input is the *instantaneous delay*. The speed of the surface is the reciprocal of instantaneous delay. The product of instantaneous delay and the speed is 1.

The block calculates the implemented transport delay by the distance-speed-time relationship, where the surface length (distance) is equal to the integral of the variable surface speed over the duration of the transport delay (time). For more information about this calculation, see Variable Transport Delay.

The Entity Transport Delay block connects SimEvents to Simulink using the input from a Simulink signal and computing the transport delay as a continuous process, and applying this delay to an entity in a discrete-event process. For an example that uses the Entity Transport Delay block, see "Modeling Cyber-Physical Systems".

Ports

Input

Port_1 — Input entity
SimEvents entity

SimEvents entity or message. For more information on entities in SimEvents, see "Entities in a SimEvents Model".

t_i — Instantaneous delay
scalar | vector | matrix

Instantaneous delay in the transport of the SimEvents entity. t_i is the reciprocal of the speed of the entity. For more information on the calculation of instantaneous delay, see "Variable Transport Delay".

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32` | `bus`

Output

Port_1 — Delayed entity
SimEvents entity

SimEvents entity with the instantaneous delay t_i applied to it.

n — Number of delayed entities
real scalar

Secondary output signal of the block, which displays the number of entities processed in a time step.

Dependencies

To enable this port, enable the **Output number of entities in block, n** parameter

Data Types: `double`

Parameters

`Capacity` — Specify the capacity of the block
inf (default) | scalar

Specify capacity to accept entities to be delayed.

Programmatic Use
Block Parameter: `Capacity`
Type: character vector
Values: `'inf'` | real scalar
Default: `'inf'`

`Output number of entities in block, n` — Outputs the number of delayed entities present in the block
off (default) | on

Number of entities present in the block that are being delayed.

Programmatic Use
Block Parameter: `ShowNumberEntitiesInBlock`
Type: character vector
Values: `'on'` | `'off'`
Default: `'off'`

Block Characteristics

Data Types	Boolean	bus	double	enumerated	fixed point	integer	single
Direct Feedthrough	no
Multidimensional Signals	yes
Variable-Size Signals	no
Zero-Crossing Detection	yes

Extended Capabilities

C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

See Also

Receive | Send | Transport Delay | Variable Transport Delay

Topics

"Establish Message Send and Receive Interfaces Between Software Components"

Introduced in R2019b

Multicast Receive Queue

Receive multicast entities
Library: SimEvents
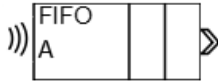
Description

The Multicast Receive Queue block is identical to an Entity Queue block with the **Entity arrival source** parameter set to `Multicast`. For information about parameter descriptions, see the documentation for the Entity Queue on page 2-51 block.

You specify a **Multicast tag** for receiving entities. Then the block receives entities with a matching **Multicast tag** broadcast by the Entity Multicast block. See "Overview of Queues and Servers in Discrete-Event Simulation", for more information about using multicast mode to broadcast entities.

Ports

Input

`receive` — **Receive broadcasted entity**
scalar | vector | matrix

Input for received entities.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `string` | `fixed point`

Output

`Port_1` — **Output entity**
scalar | vector | matrix

Output entity at the head of the queue to depart when a downstream block is ready to accept it.

Note Event actions are not supported with string entity data type

---

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `string` | `fixed point`

See Also

Entity Selector | Entity Store | Entity Generator | Entity Multicast | Entity Server

Topics
"Overview of Queues and Servers in Discrete-Event Simulation"
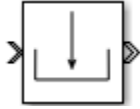"Storage with Queues"
"SimEvents Common Design Patterns"

Introduced in R2016a

Resource Acquirer

Acquire entity resources
Library: SimEvents


Description

The Resource Acquirer block acquires for and assigns them to input entities. You can specify which resource amounts and type the block acquires.

An entity does not depart the Resource Acquirer block until the entity acquires all of the requested resources. For example, if an entity requests 5 resources and there are 2 resources available in a Resource Pool block, then the entity waits until all the requested resources are available before exiting. Similarly, if an entity requests 2 resources from one type and 3 resources from another type, the entity waits until all of the resources from both types are available.

Initialize a Resource Pool block with a specified amount of available resources. Then:

• Use one or more Resource Acquirer blocks to reserve those resources.

  The priority order of Resource Acquirer blocks is determined at the beginning of a simulation and cannot be customized. The entity in the higher priority Resource Acquirer block always acquires the resource first.
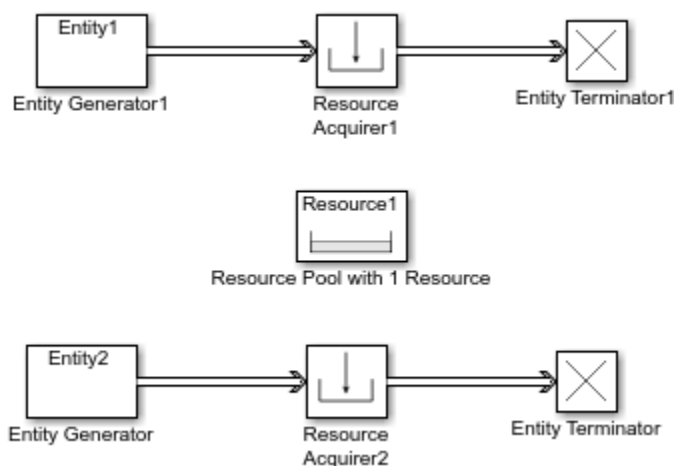
For example, suppose only 1 resource is available in the Resource Pool block and Resource Acquirer1 is higher priority than Resource Acquirer2. If Entity1 and Entity2 want to acquire the resource at the same time, Entity1 always acquires the resource. Even if the resource becomes available again and there are two entities Entity1 and Entity2 waiting in the ResourceAcquirer1 and ResourceAcquirer2, Entity1 again acquires the resource.

• Use a Resource Releaser block to return resources to the Resource Pool block for future use.

The visibility of the resources is determined by the **Resource visibility** parameter of the Resource Pool blocks in the model.

To customize actions when entities enter, exit, or are blocked, enter MATLAB code in the `Entry action`, `Exit action`, or `Blocked action` field of the **Event actions** tab.

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 **Available Resources** list. The list updates with any upstream model changes you make while the block dialog box is open.
⟼	Add the selected resources to the **Selected Resources** table.
⟻	Move the selected resource from the **Selected Resources** table to the **Available Resources** 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 **Selected Resources** 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 **Selected Resources** table.	When you delete a resource this way, no confirmation appears and you cannot undo the operation.
	Move up the selected resource in order in the **Selected Resources** table.	N/A
	Move the selected resource down in order in the **Selected Resources** 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. Each resource must have a unique name.	Double-click the existing name, and then type the new name.
Amount Source	Whether the resource amount, that an entity requests, comes from the dialog box or an attribute.	Select `Dialog` or `Attribute`. If you select `Attribute`, 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"

Property	Specify	Use
Amount	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 **Amount** is 3, each entity that arrives at the Resource Acquire block must wait to acquire 3 resources before departing the block. Granularity of the resources to be acquired matches the granularity of the resources in the Resource Pool block.

Ports

Input

Port_1 — Input entity
scalar | vector | matrix

Input entity port for entities entering the block.

Data Types: single | double | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64 | Boolean | enumerated | bus | fixed point

Output

Port_1 — Output entity
scalar | vector | matrix

Output entity port for entities exiting the block.

Data Types: single | double | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64 | Boolean | enumerated | bus | fixed point

Port_d — Number of entities that have departed the block
scalar

Number of entities that have departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities departed, d**.

Data Types: double

Port_n — Number of entities that have not yet departed the block
scalar

Number of entities that have not yet departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities in block, n**.

Data Types: `double`

Port_w — Average wait time for entities in the block
scalar

Average wait time for entities in the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Average wait, w**.

Data Types: `double`

Port_ex — Number of entities extracted
scalar

Number of entities that are pulled out of this block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities extracted, ex**.

Data Types: `double`

Parameters

Maximum number of waiting entities — Maximum number of entities that can wait for a resource
1 (default) | `scalar`

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

Programmatic Use
Block Parameter: NumberWaitingEntities
Type: character vector
Values: `'1'` | scalar
Default: `'1'`

Event actions — Specify the behavior of the entity on certain events
Entry (default) | Exit | Blocked

Define the behavior in the **Event actions**. For example, the **Entry** action is called when an entity enters the block.

Programmatic Use
Block Parameter: EntryAction, ExitAction, BlockedAction
Type: character vector
Values: MATLAB code
Default: `''`

Number of entities departed, d — Outputs the number of entities that have departed the block
off (default) | on

Selecting this check box outputs the number of entities that have exited the block.

Programmatic Use
Block Parameter: NumberEntitiesDeparted
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities in block, n — Outputs the number of entities present in the block, which have yet to depart
off (default) | on

Selecting this check box outputs the number of entities present in the block, which have yet to depart.

Programmatic Use
Block Parameter: NumberEntitiesInBlock
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Average wait, w — Outputs the average wait time
off (default) | on

Sum of the wait times for entities departing the block divided by their total number. Wait time is the duration between the entity's entry into and exit from the Resource Acquirer block. For more information, see "Interpret SimEvents Models Using Statistical Analysis".

Programmatic Use
Block Parameter: AverageWait
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities extracted, ex — Number of entities extracted from this block
off (default) | on

Outputs the number of extracted entities which are pulled out from this block by the Entity Find block. When an entity is extracted, its resource acquisition from the Resource Pool block is canceled and **Number of entities departed, d**, **Number of entities in block, n**, and **Average wait, w** statistics are updated accordingly. For more information about finding and extracting entities, see "Find and Extract Entities in SimEvents Models".

Programmatic Use
Block Parameter: NumEntitiesExtracted
Type: character vector
Values: 'on' | 'off'
Default: 'off'

See Also

Entity Generator | Resource Pool | Resource Releaser

Topics
"Model Using Resources"
"SimEvents Common Design Patterns"

Introduced in R2016a

Resource Pool

Pool entity resources

Library: SimEvents / Entity Management
SimEvents

Description

The Resource Pool 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 those resources.

- Use a Resource Releaser block to return resources back to this block for future use.

You can determine the visibility of available resources in a model hierarchy. You can choose `Global` or `Scoped` resources in the pool.

- `Global` — Resources can be referenced from anywhere in a model hierarchy.

- `Scoped` — Resources are locally visible and can be referenced only from the subsystem that contains the Resource Pool block and all the subsystems inside.

Ports

Input

Port_1 — Change resource amount
scalar | vector | matrix

Input entity port for changing resource amount. The input cannot be a negative value.

Dependencies

To enable this port, select the `Change amount through control port` as the **Resource amount source**.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `fixed point`

Output

Port_u — Outputs the number of resources that are in use
scalar

Number of resources that are in use.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Amount in use, u**.

Data Types: `double`

Port_util — Outputs the average time the pool is utilized
scalar

Average wait time the pool is utilized.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Average utilization, util**.

Data Types: `double`

Port_avail — Outputs the number of resources that are available
scalar

Number of resources that are available.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Amount available, avail**.

Data Types: `double`

Parameters

Resource name — Name for the resources in the pool
`Resource1` (default) | character vector

Enter name of entity resource.

Programmatic Use
Block Parameter: `ResourceName`
Type: character vector
Values: `'Resource1'` | character vector
Default: `'Resource1'`

Resource granularity — Select granularity of resource use
`Discrete unit` (default) | `Fractional amount`

Select granularity of resource use.

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

Programmatic Use
Block Parameter: `ResourceGranularity`
Type: character vector
Values: `'Discrete unit'` | `'Fractional amount'`
Default: `'Discrete unit'`

Reusable upon release — Specify if the resource is reusable upon release
`off` (default) | `on`

- 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.

- Clear this check box to prevent this resource from returning to the resource pool upon release. In this case, when the resource is released, it is no longer available in the resource pool. An example of such a resource is food in a restaurant, which is not reusable upon consumption.

Programmatic Use
Block Parameter: ReusableUponRelease
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Resource amount source — Select resource amount source
Dialog (default) | Change amount through control port

Select resource amount source.

- Dialog

- Change amount through control port

  Select this option to enable an input entity port and a variable capacity resource. The payload of the arriving message increments the existing number of resources for the block. For example, if the resource pool has five resources, and a message with a payload of three arrives at the input port, the block has eight resources available. The number of resources cannot decrement.

Programmatic Use
Block Parameter: ResourceAmountSource
Type: character vector
Values: 'Dialog' | 'Change amount through control port'
Default: 'Dialog'

Resource amount — Set the amount of resource
10 (default) | scalar

Enter amount of resource.

Dependencies

Select the Dialog to enable the **Resource amount source**.

Programmatic Use
Block Parameter: ResourceAmount
Type: character vector
Values: '10' | scalar
Default: '10'

Initial resource amount — Enter initial amount of resource
10 (default) | scalar

Enter initial amount of resource.

Dependencies

Select the Change amount through control port to enable the **Resource amount source**.

Programmatic Use
Block Parameter: InitialResourceAmount
Type: character vector
Values: '10' | scalar
Default: '10'

Resource visibility — Select the availability of the resources
Global (default) | Scoped

Choose the behavior of the resources acquired from this pool as Global or Scoped.

When Global is selected:

- Resource pool names must be unique in the model.
- All resources have global scope and they can be referenced from anywhere in a model hierarchy.
- An entity carrying a resource acquired from this block, must explicitly relinquish the resource.
- When an entity is destroyed, the resources that are associated with it are returned to the pool, if the **Reusable upon release** check box is selected.

When Scoped is selected:

- Resources are locally visible and can be referenced only from the subsystem that contains the Resource Pool block and all the subsystems inside.
- Resource pool names must be unique within the model hierarchy where the Resource Pool block is visible.
- An entity carrying the resource acquired from this block, must explicitly relinquish the resource.
- When an entity leaves the scope, any resources that are local to that scope and not already released, are automatically released. If **Reusable upon release** check box is selected, they are returned to the pool.

Programmatic Use
Block Parameter: InitialResourceAmount
Type: character vector
Values: '10' | scalar
Default: '10'

Amount in use, u — Number of resources that are in use
off (default) | on

Outputs the amount of resources that the block has acquired and has not yet released. For example, if the resource pool has 10 resources, and the entity acquires all of them, this port shows 10. When the block releases the resources, this port shows 0.

Programmatic Use
Block Parameter: AmountInUse
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Average utilization, util — Outputs the average resource utilization
off (default) | on

Outputs the average resource utilization.

Programmatic Use
Block Parameter: AverageUtilization
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Amount available, avail — Outputs the amount of resources available
off (default) | on

Outputs the amount of resources available.

Programmatic Use
Block Parameter: AmountAvailable
Type: character vector
Values: 'on' | 'off'
Default: 'off'

See Also

Entity Generator | Resource Acquirer | Resource Releaser
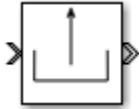
Topics
"Model Using Resources"
"SimEvents Common Design Patterns"

Introduced in R2016a

Resource Releaser

Release entity resources
Library: SimEvents

Description

The Resource Releaser block releases resources when an entity enters the block. The block accepts one entity and the entity departs the block immediately provided it is accepted by the next block or extracted by the Entity Find block.

You can specify that the block release certain resource types or release all resources.

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 **Available Resources** list. The list updates with any upstream model changes you make while the block dialog box is open.
	Add the selected resources to the **Selected Resources** table.
	Move the selected resource from the **Selected Resources** table to the **Available Resources** list. 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 **Selected Resources** 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.
- Choose the amount of resources to be released by setting the **Amount Source** parameter to `Dialog` or `Attribute`.

  - `Dialog` — Specify the amount of resources to be released under the **Amount** column.
  - `Attribute` — Specify the name of the attribute that defines the amount of resources to be released.

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 **Selected Resources** table.	When you delete a resource this way, no confirmation appears and you cannot undo the operation.
⬆	Move up the selected resource in order in the **Selected Resources** table.	N/A
⬇	Move the selected resource down in order in the **Selected Resources** 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	Specification	Usage
Name	The name of the resource. Each resource must have a unique name.	Double-click the existing name, and then type the new name.
Amount Source	Whether the resource amount, that an entity requests, comes from the dialog box or an attribute.	Select `Dialog` or `Attribute`. If you select `Attribute`, 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".
Amount	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 released per entity. For example, if **Amount** is three, each entity that arrives at the Resource Releaser block must wait to release 3 resources before departing the block. Granularity of the resources to be released matches the granularity of the resources in the Resource Pool block.

Ports

Input

Port_1 — Input entity
scalar | vector | matrix

Input entity port for entities entering the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Output

Port_1 — Output entity
scalar | vector | matrix

Output entity port for entities exiting the block.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `Boolean` | `enumerated` | `bus` | `fixed point`

Port_d — Number of entities that have departed the block
scalar

Number of entities that have departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities departed, d**.

Data Types: `double`

Port_n — Number of entities that have not yet departed the block
scalar

Number of entities that have not yet departed the block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities in block, n**.

Data Types: `double`

Port_ex — Number of entities extracted
scalar

Number of entities that are pulled out of this block.

Dependencies

To enable this port, click the **Statistics** tab and select the box labeled **Number of entities extracted, ex**.

Data Types: `double`

Parameters

Resource to release — Select the resources to release
`All` (default) | `Selected`

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.

Programmatic Use
Block Parameter: `ResourceToRelease`
Type: character vector
Values: `'All'` | `'Selected'`
Default: `'All'`

Event actions — Specify the behavior of the entity on certain events
`Entry` (default) | `Exit` | `Blocked`

Define the behavior in the **Event actions**. For example, the **Entry** action is called when an entity enters the block.

Programmatic Use
Block Parameter: EntryAction, ExitAction, BlockedAction
Type: character vector
Values: MATLAB code
Default: ''

Number of entities departed, d — Outputs the number of entities that have departed the block
off (default) | on

Selecting this check box outputs the number of entities that have exited the block.

Programmatic Use
Block Parameter: NumberEntitiesDeparted
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities in block, n — Outputs the number of entities present in the block, which have yet to depart
off (default) | on

Selecting this check box outputs the number of entities present in the block, which have yet to depart.

Programmatic Use
Block Parameter: NumberEntitiesInBlock
Type: character vector
Values: 'on' | 'off'
Default: 'off'

Number of entities extracted, ex — Number of entities extracted from this block
off (default) | on

Number of extracted entities that are pulled out from this block by the Entity Find block. For more information about finding and extracting entities, see "Find and Extract Entities in SimEvents Models".

Programmatic Use
Block Parameter: NumEntitiesExtracted
Type: character vector
Values: 'on' | 'off'
Default: 'off'

See Also

Entity Generator | Resource Acquirer | Resource Pool

Topics
"Model Using Resources"
"SimEvents Common Design Patterns"

Introduced in R2016a

Sequence Viewer

Display messages, events, states, transitions, and functions between blocks during simulation

Library: Simulink / Messages & Events
Simulink Test
SimEvents
Stateflow

Description

The Sequence Viewer block displays messages, events, states, transitions, and functions between certain blocks during simulation. The blocks that you can display are called lifeline blocks and include:

- Subsystems
- Referenced models
- Blocks that contain messages, such as Stateflow charts
- Blocks that call functions or generate events, such as Function Caller, Function-Call Generator, and MATLAB Function blocks
- Blocks that contain functions, such as Function-Call Subsystem and Simulink Function blocks

To see states, transitions, and events for lifeline blocks in a referenced model, you must have a Sequence Viewer block in the referenced model. Without a Sequence Viewer block in the referenced model, you can see only messages and functions for lifeline blocks in the referenced model.

Parameters

`Time Precision for Variable Step` — Digits for time increment precision
3 (default) | scalar

Number of digits for time increment precision. When using a variable step solver, change this parameter to adjust the time precision for the sequence viewer. By default the block supports 3 digits of precision.

Suppose the block displays two events that occur at times `0.1215` and `0.1219`. Displaying these two events precisely requires 4 digits of precision. If the precision is 3, then the block displays two events at time `0.121`.

Programmatic Use
Block Parameter: `VariableStepTimePrecision`
Type: character vector
Values: `'3'` | scalar
Default: `'3'`

`History` — Maximum number of previous events to display
5000 (default) | scalar

Total number of events before the last event to display.

For example, if **History** is 5 and there are 10 events in your simulation, then the block displays 6 events, including the last event and the five events prior the last event. Earlier events are not displayed. The time ruler is greyed to indicate the time between the beginning of the simulation and the time of the first displayed event.

Each send, receive, drop, or function call event is counted as one event, even if they occur at the same simulation time.

Programmatic Use
Block Parameter: `History`
Type: character vector
Values: `'1000'` | scalar
Default: `'1000'`

Block Characteristics

Data Types	`Boolean` | `bus` | `double` | `enumerated` | `fixed point` | `integer` | `single`
Direct Feedthrough	`no`
Multidimensional Signals	`yes`
Variable-Size Signals	`no`
Zero-Crossing Detection	`no`

Extended Capabilities

C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

This block can be used for visualizing message transitions during simulation, but is not included in the generated code.

HDL Code Generation
Generate Verilog and VHDL code for FPGA and ASIC designs using HDL Coder™.

This block displays messages during simulation when used in subsystems that generate HDL code, but it is not included in the hardware implementation.

See Also
"Use the Sequence Viewer to Visualize Messages, Events, and Entities"

Introduced in R2015b

2-121

SimEvents Debugger

Debug SimEvents models
Library: SimEvents

Description

The SimEvents Debugger block enables the debugger for your SimEvents model. Using this block, 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, add the SimEvents Debugger block into your SimEvents model.

2 In the Simulink Editor, click the **Step Forward** button.

The debugger interface appears.
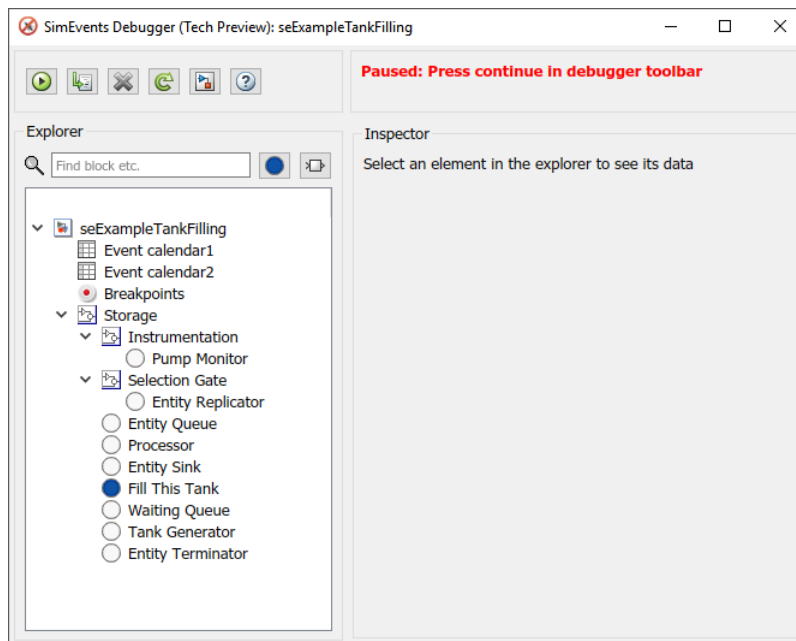
- Click the **Continue simulation** button ( ) to begin the simulation in the debugger. When the simulation completes, the debugger interface closes.

- 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 `tnow`.

- 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.

Parameters

Enable debugger — Enable the debugger
on (default) | off

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

See Also

Entity Generator | Entity Queue

Topics
"Debug SimEvents Models"
"Simulation Stepper"
"Visualization and Animation for Debugging"

Introduced in R2016a

Attribute Function (Obsolete)

Access and modify attributes using MATLAB code

Library

Attributes

Description

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.

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 **Switching criterion**=From attribute
Preemption based on the same attribute that the Attribute Function block created or modified	Single Server block with **Permit preemption based on attribute** 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"

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

SimEvents Sinks

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** 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 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 **Enable entity OUT port**.

Signal Output Ports

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

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

Parameters

Plotting Tab

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 2-127 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 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 attribute 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.

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`.

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**.

Examples

- "Set Attributes"

See Also

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

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.

2-131

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 **rt** 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

Parameters

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 from the action bar, select the subsystem. 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.

Parameters

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
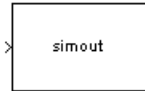
Library

SimEvents Sinks

Description

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 **Pause** under the **Debug** tab. 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 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.

Parameters

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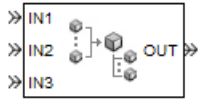
Allows you to use the signal output port labeled **#d**.

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

☑ Retain structure in departing entity ☑ Make attributes accessible in departing entity ☑ Make timers accessible in departing entity • You can split the departing entity, which is called a composite entity. • Attribute names and timer tags must be unique. • You can access attributes and timers.	☐ Retain structure in departing entity ☑ Copy attributes to departing entity ☑ Copy timers to departing entity • You cannot split the departing entity. • Attribute names and timer tags must be unique. • You can access attributes and timers.
☑ Retain structure in departing entity ☐ Make attributes accessible in departing entity ☐ Make timers accessible in departing entity • 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. • Uniqueness of attribute names and timer tags is optional. • You cannot access attributes and timers via the departing entity.	☐ Retain structure in departing entity ☐ Copy attributes to departing entity ☐ Copy timers to departing entity • You cannot split the departing entity. • Uniqueness of attribute names and timer tags is optional. • You cannot access attributes and timers.

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 **Number of entity input ports** 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.

Parameters

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)

"Model Resource Allocation Using Composite Entity Creator block"

Introduced in R2007a

Entity Departure Counter (Obsolete)

Count departures and write result to signal port or attribute

Library

Entity Management

Description

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 **#d** output signal to zero. This signal must be an event-based signal. You see this port only if you set **Reset counter upon** to `Sample time hit from port ts`.
tr	When this signal satisfies the specified trigger criteria, the block resets its internal counter and the **#d** output signal to zero. This signal must be an event-based signal. You see this port only if you set **Reset counter upon** to `Trigger from port tr`.
vc	When this signal satisfies the specified value-change criteria, the block resets its internal counter and the **#d** output signal to zero. This signal must be an event-based signal. You see this port only if you set **Reset counter upon** to `Change in signal from port vc`.
fcn	When this signal carries a function call, the block resets its internal counter and the **#d** output signal to zero. This signal must be an event-based function call. You see this port only if you set **Reset counter upon** to `Function call 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 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.

Parameters

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

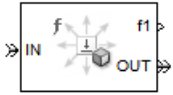- "Set Attributes"

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

Description

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

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, possibly contingent on **e1** input signal.	Before or after entity departure, depending on **Timing of function call f1** parameter	1
f2	Function-call signal, possibly contingent on **e2** input signal.	Before entity departure if both **Timing of function call...** parameters are set to `Before entity departure`; otherwise, after entity departure	2
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	4
#f1	Number of function calls the block has generated at the **f1** port during the simulation.	After entity departure	3
#f2	Number of function calls the block has generated at the **f2** 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.

Parameters

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

Entity Sink (Obsolete)

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 **Input port available for entity arrivals**, and then select the **Report number of entities arrived, #a** 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.

Parameters

Input port available for entity arrivals

Determines whether the block accepts or blocks entities that attempt to arrive.
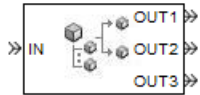
Report number of entities arrived, #a

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.

Parameter Value	Description
`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 **OUT1**, **OUT2**, **OUT3**,..., in that sequence.	The sequence of departures is always **OUT1**, **OUT2**, **OUT3**,... 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 **OUT1**, **OUT2**, **OUT3**. The second time, the component entities depart in the sequence **OUT2**, **OUT3**, **OUT1**. The third time, the component entities depart in the sequence **OUT3**, **OUT1**, **OUT2**. 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 **Initial seed** 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 **OUT3**, **OUT4**, **OUT1**, **OUT2**. If the random number is two on the next such occasion, then the component entities depart in the sequence **OUT2**, **OUT3**, **OUT4**, **OUT1**.

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.

Parameter Value	Description
`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 Entity Combiner block using the **Retain structure in departing entity** 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 **OUT***N* port corresponds to the entity that arrived at the **IN***N* entity input port of the corresponding Entity Combiner block. The **Number of entity output ports** 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.

Parameters

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)

"Model Resource Allocation Using Composite Entity Creator block"

Introduced in R2007a

Entity-Based Function-Call Event Generator (Obsolete)

Generate function call events corresponding to entities

Library

Generators / Event Generators

Description

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 **Generate function call** 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.

Parameters

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 Filter (Obsolete)

Conditionalize, suppress, or prioritize execution of Atomic Subsystem

Library

SimEvents Ports and Subsystems

Description

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 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 Atomic Subsystem block. When the Event Filter 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.

Parameters

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 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

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-Based Entity Generator (Obsolete)

Generate entity upon signal-based event or function call

Library

Generators / Entity Generators

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 ts`
Each time an input signal has a trigger edge	`Trigger from port tr`
Each time an input signal changes its value	`Change in signal from port vc`
Each time an input signal carries a function call	`Function call from port fcn`

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 2-162.

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 **Allow OUT port blocking** check box.
The block stores the entity as a pending entity, and immediately discards it. The entity does not depart from the block.	Select **Allow OUT port blocking** and set **Response during blockage period** to `Discard generated entities`

Response to Blockage	Parameter Values
The block stores the entity as a pending entity, and temporarily suspends the generation of additional entities. During this suspension, when the block executes `EntityGeneration` 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 **Allow OUT port blocking** and set **Response during blockage period** to `Delay first pending entity`

Ports

Signal Input Ports

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

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	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 or discarding of the pending entity.	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

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.

Parameters

Entity Generation Tab

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. You see this field only if you set **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. You see this field only if you set **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. If you do not select this option, the event has priority SYS1 on the event calendar.

Generation event priority

The priority of the entity-generation 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**.

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. You see this field only if you set **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. You see this field only if you set **Generate entities upon** to `Function call from port fcn`.

Blocking Tab

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 2-162. You see this field only if you select **Allow OUT port blocking**.

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

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**.
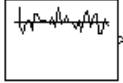
See Also

Time-Based Entity Generator (Obsolete)

Entity Sink (Obsolete)

"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.

Distribution	Additional Parameters
Exponential	**Mean**
Uniform	**Minimum**, **Maximum**

Distribution	Additional Parameters
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 2-172 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.

Parameters

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

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

where μ is the **Mean** parameter, a positive number.

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

Uniform Distribution

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

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

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

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

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

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

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

where θ is the **Threshold** parameter, b is the **Scale** parameter, and a is the **Shape** parameter. The **Scale** and **Shape** parameters must be positive. Also, Γ is the gamma function (`gamma` in MATLAB code).

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

Gaussian (Normal) Distribution

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

where μ is the **Mean** parameter and σ 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

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

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

$$f(x) = \begin{cases} \dfrac{e^{-\lambda}\lambda^x}{x!} & \text{for } x = 0, 1, 2, \ldots \\ 0 & \text{otherwise} \end{cases}$$

where λ 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

$$f(x) = \begin{cases} \dfrac{\exp\left[\dfrac{-(\ln(x-\theta)-\mu)^2}{2\sigma^2}\right]}{(x-\theta)\sigma\sqrt{2\pi}} & \text{for } x \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where θ is the **Threshold** parameter, μ is the **Mu** parameter, and σ 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 θ is the **Threshold** parameter and b is the **Scale** parameter. The **Scale** parameter must be positive.

Beta Distribution

$$f(x) = \begin{cases} \dfrac{(x-L)^{a-1}(U-x)^{b-1}}{B(a,b)(U-L)^{a+b+1}} & \text{for } L \leq x \leq U \\ 0 & \text{otherwise} \end{cases}$$

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

$$f(x) = \begin{cases} 1/K & \text{for } x = L + k\dfrac{(U-L)}{K-1}, \ k = 0, 1, 2, ..., K-1 \\ 0 & \text{otherwise} \end{cases}$$

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

$$f(x) = \begin{cases} \dfrac{\gamma}{\alpha}\left(\dfrac{x-\theta}{\alpha}\right)^{(\gamma-1)}\exp\left[-\left(\dfrac{x-\theta}{\alpha}\right)^{\gamma}\right] & \text{for } x \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where θ is the **Threshold** parameter, α is the **Scale** parameter, and γ 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.

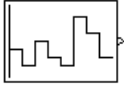[4] Johnson, N. L., S. Kotz, and A. W. Kemp. *Univariate Discrete Distributions*. Wiley-Interscience, 1993.

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.
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 inf 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.

Parameters

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

FIFO Queue (Obsolete)

Store entities in sequence for undetermined length of time

Library

Queues

Description

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.
TO	Port for entities that time out while in this block. You see this port only if you select **Enable TO port for timed-out entities**. 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 **OUT** port since the start of the simulation.	After entity departure via the **OUT** port	3
#n	Number of entities currently in the queue.	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.	After entity departure	1
len	Average number of entities in the queue over time, that is, the time average of the **#n** signal.	After entity arrival and 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

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.

Parameters

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
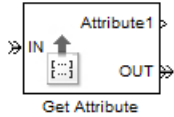
- "Create a Discrete-Event Model"

See Also

LIFO Queue (Obsolete), Priority Queue (Obsolete)

"Overview of Queues and Servers in Discrete-Event Simulation"

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.

Parameters

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 **Available Attributes** 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 **Get Attribute** table.
	Move the selected attribute from the **Get Attribute** table to the **Available Attributes** 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 **Get Attribute** 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 **Get Attribute** table, the behavior depends on how the **If Missing** 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 **Get Attribute** 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
Attribute Name	The name of the attribute to access. Each attribute must have a unique name. If the attribute name does not match an attribute listed in **Available Attributes**, the block displays **??** next to the attribute name. This symbol denotes that the attribute is not present on incoming entity paths. The simulation behavior you see depends on the value of **If Missing** (see "Missing Attributes" on page 2-180).	Double-click the existing name, and then type the new name.
If Missing	The response of the block when the entity does not have an attribute named in the table.	Select a block response from the list.
Default Value	The value for the corresponding output signal if the entity does not have an attribute specified in the table. You can set this field only if you set **If Missing** to `Default value` or `Warn`.	Double-click the field and type a value.
Vector Is 1-D	Whether the block considers the default value as a vector of length N when **Default Value** 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 **If Missing** parameter is set to `Default value` or `Warn`.	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 **Default Value** and **Treat Vector as 1-D** parameters are disabled.
`Default value`	The block outputs a default value that you specify using the **Default Value** and **Treat Vector as 1-D** parameters. The simulation proceeds.

Parameter Value	Block Behavior in Case of Missing Attribute
Warn	The block outputs a default value that you specify using the **Default Value** and **Treat Vector as 1-D** 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
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure
Attribute name	Value of the attribute of the same name specified in the table. The default **Name** that corresponds to each row is **Attribute*x***, 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)

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 **Service time from** to `Signal port t`.

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 **Enable TO port for timed-out entities**. 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 **OUT** port since the start of the simulation.	After entity departure via the **OUT** 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. 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 entity's waiting time might exceed its service time if the **OUT** port is blocked when the entity completes service.	After entity departure	2
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the **TO** 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.

Parameters

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)

Introduced before R2006a

Initial Value (Obsolete)

Output specified value until first sample time hit

Library

Signal Management

Description

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.
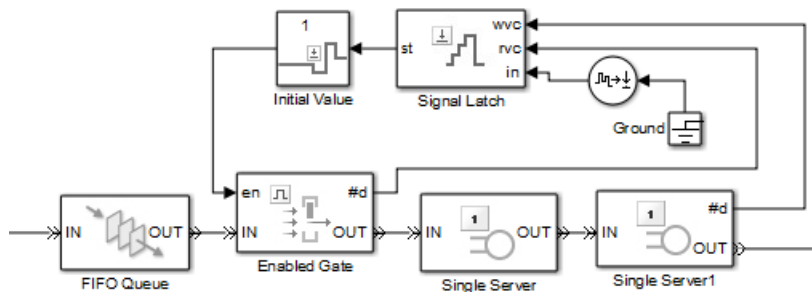
Note The IC block in the Simulink library set operates in a time-based manner and is not suitable for event-based signals.

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.

Parameters

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

Description

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, **IN1** 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, **IN1**.
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 **Initial seed** parameter initializes the random number generation process.
From signal port p	Selecting this option creates an additional signal input port, labeled **p**. The signal at this port must have integer values between 1 and the **Number of entity input ports** 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 **Number of entity input ports** 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,..., **Number of entity input ports**. This signal must be an event-based signal. You see this port only if you set **Switching criterion** to `From signal port p`.

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	2
last	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,..., **Number of entity input ports**.	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.

Parameters

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

Description

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 **Enable entity OUT port**.

Parameters

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
ts	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 **Plot points upon** to `Sample time hit from port ts`.
tr	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 **Plot points upon** to `Trigger from port tr`.
vc	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 **Plot points upon** to `Change in signal from port vc`.
fcn	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 **Plot points upon** to `Function call from port fcn`.

Parameters

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.

2-195

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

Description

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.
TO	Port for entities that time out while in this block. You see this port only if you select **Enable TO port for timed-out entities**. 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 **OUT** port since the start of the simulation.	After entity departure via the **OUT** port	3
#n	Number of entities currently in the queue.	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.	After entity departure	1
len	Average number of entities in the queue over time, that is, the time average of the **#n** signal.	After entity arrival and 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

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.

Parameters

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

Description

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 **Service time from** to `Signal port t`.
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 **Allow service control** and set **Service change upon disabling** to `Pause`.
complete	Port for input signal that disables all servers when the signal is positive. 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. You see this port only if you select **Allow service control** and set **Service change upon disabling** to `Force complete`.

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.
TO	Port for entities that time out while in this block. You see this port only if you select **Enable TO port for timed-out entities**. 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 **OUT** port since the start of the simulation.	After entity departure via the **OUT** 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 entity's waiting time might exceed its service time if the **OUT** port is blocked when the entity completes service.	After entity departure	2
util	Utilization of the N-server. If **Number of servers** is finite, **util** is the time average of the fraction of servers that are storing an entity. At time values when an entity arrives or departs, **util** equals 1/N times the time average of the **#n** signal. If **Number of servers** is infinite, then **util** is always zero.	Performance considerations cause the block to update the signal only after each arrival or departure of an entity.	2
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the **TO** port	5

Port Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
so	Occupancy status of each server in the N-Server block. The **so** port outputs a vector of values. If a server is unoccupied, the value of the corresponding vector element is **0**. If a server is occupied, the vector element has a value of **1**.	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 kth 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 $\#n/N$.

Parameters

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

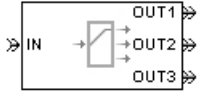Allows you to use the signal output port labeled **so**.

See Also

Single Server (Obsolete), Infinite Server (Obsolete)

Introduced before R2006a

Output Switch (Obsolete)

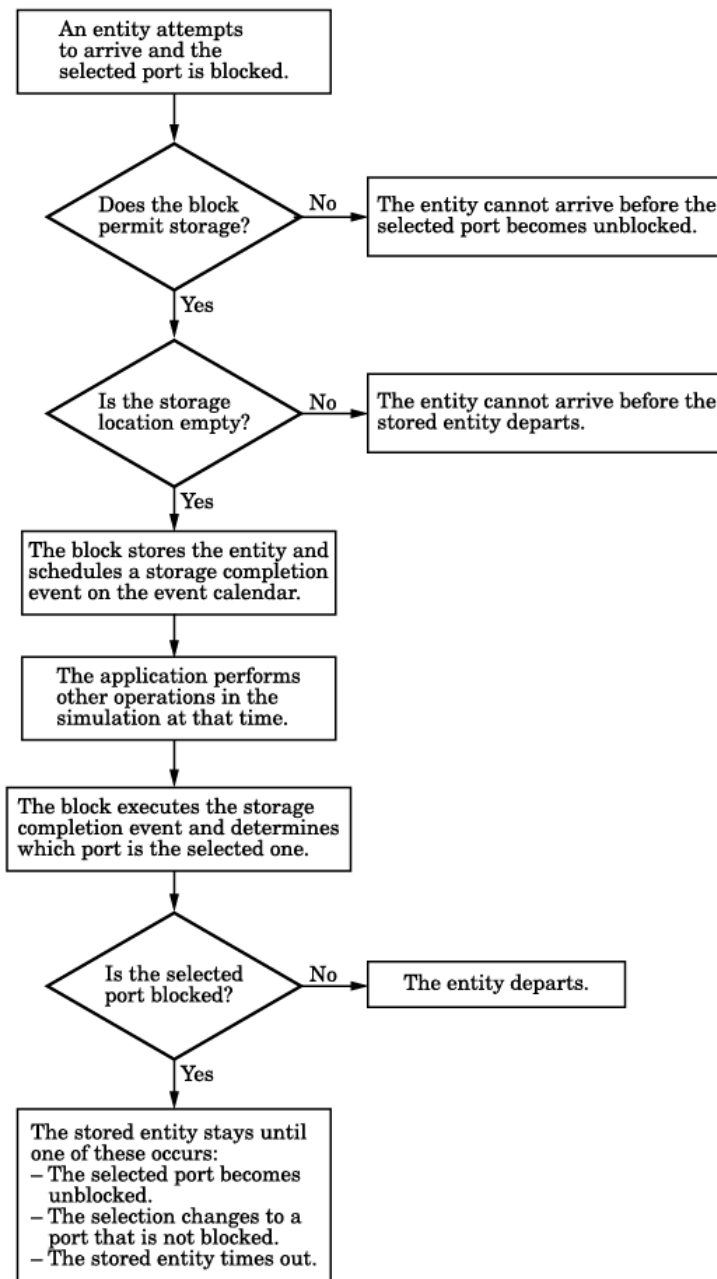Select entity output port for departure

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.

```
┌─────────────────────┐
│  An entity attempts │
│  to arrive and the  │
│ selected port is blocked. │
└─────────────────────┘
          │
          ▼
      ╱─────────╲        No    ┌─────────────────────────┐
     ╱ Does the  ╲─────────────▶│ The entity cannot arrive before the │
     ╲ block      ╱             │ selected port becomes unblocked. │
      ╲permit storage?╱         └─────────────────────────┘
          │
         Yes
          │
          ▼
      ╱─────────╲        No    ┌─────────────────────────┐
     ╱ Is the    ╲─────────────▶│ The entity cannot arrive before the │
     ╲ storage    ╱             │ stored entity departs. │
      ╲location empty?╱         └─────────────────────────┘
          │
         Yes
          │
          ▼
┌─────────────────────┐
│ The block stores the entity and │
│ schedules a storage completion │
│ event on the event calendar. │
└─────────────────────┘
          │
          ▼
┌─────────────────────┐
│ The application performs │
│ other operations in the │
│ simulation at that time. │
└─────────────────────┘
          │
          ▼
┌─────────────────────┐
│ The block executes the storage │
│ completion event and determines │
│ which port is the selected one. │
└─────────────────────┘
          │
          ▼
      ╱─────────╲        No    ┌─────────────────────────┐
     ╱ Is the selected ╲───────▶│ The entity departs. │
     ╲ port blocked?   ╱        └─────────────────────────┘
          │
         Yes
          │
          ▼
┌─────────────────────┐
│ The stored entity stays until │
│ one of these occurs: │
│ – The selected port becomes │
│   unblocked. │
│ – The selection changes to a │
│   port that is not blocked. │
│ – The stored entity times out. │
└─────────────────────┘
```

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 **OUT1** 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, **OUT1**.
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 **Initial seed** 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 **OUT1**. If that port is blocked, then the block attempts to output the entity through **OUT2**, and so on. If all entity output ports are blocked, then this block's **IN** port is unavailable and the entity cannot arrive.
From signal port p	Selecting this option creates an additional signal input port, labeled **p**. The signal at this port uses integer values between 1 and the **Number of entity output ports** parameter value to refer to entity output ports. The block monitors the **p** signal's value throughout the simulation and reacts to changes by selecting the corresponding entity output port.
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 **Attribute name** parameter. The attribute value must be an integer between 1 and the **Number of entity output ports** 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 "Control Output Switch with Event Actions and Simulink Function".

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 **Number of entity output ports**. This signal must be an event-based signal. You see this port only if you set **Switching criterion** to `From signal port p`.

Entity Output Ports

Label	Description
OUT1, **OUT2**, **OUT3**, and so on	Entity ports through which an arriving entity departs, where the **Switching criterion** parameter determines which of multiple ports the entity departs through. The **Number of entity output ports** 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 **Switching criterion** to `From signal port p`, select **Store entity before switching**, and select **Enable TO port for timed-out entities**. 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
#d	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 **TO**	3
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the **TO** port	2
pe	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
last	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,..., **Number of entity output ports**.	After entity departure via a port other than **TO**	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.

Parameters

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)

"Control Output Switch with Event Actions and Simulink Function"

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 **IN1**, **IN2**, **IN3**,... until either an entity arrives or all ports are notified.	Throughout the simulation, the sequence of notifications is always **IN1**, **IN2**, **IN3**, **IN4**.
Equiprobable	Notify a random entity input port. All are equally likely and the **Initial seed** 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 **IN3**, **IN4**, **IN1**, **IN2**. If the random number is two on the next such occasion, notify the ports in the sequence **IN2**, **IN3**, **IN4**, **IN1**.

Input Port Precedence	Action when Entity Output Port Becomes Unblocked	Example for Block with Four Entity Input Ports
Round robin	Notify the port next to the one through which the last departing entity arrived. The **IN1** 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 **IN2** port and advances to a Single Server block. Meanwhile, entities attempt to arrive at the Path Combiner block. When the server becomes available, the Path Combiner block notifies the ports in the sequence **IN3**, **IN4**, **IN1**, **IN2**. The sequence starts with **IN3** because it is next to **IN2**, which is the port through which the last departing entity arrived.
From signal port p	Notify the port whose index is the value of the **p** 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 **p** signal is three, notify the ports in the sequence **IN3**, **IN4**, **IN1**, **IN2**. If **p** is two on the next such occasion, notify the ports in the sequence **IN2**, **IN3**, **IN4**, **IN1**.

Ports

Entity Input Ports

Label	Description
IN1, **IN2**, **IN3**, and so on	Port for arriving entities. The **Number of entity input ports** 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 the block makes available first, upon an event that changes the entity output port from blocked to unblocked. Values are 1, 2, 3,..., **Number of entity input ports**. This signal must be an event-based signal. You see this port only if you set **Input port precedence** to From signal port p.

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	2
last	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,..., **Number of entity input ports**.	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.

Parameters

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

Description

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
OUT	Port for departing entities that do not time out while in this block.
TO	Port for entities that time out while in this block. You see this port only if you select **Enable TO port for timed-out entities**. 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 **OUT** port since the start of the simulation.	After entity departure via the **OUT** port	3
#n	Number of entities currently in the queue.	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.	After entity departure	1
len	Average number of entities in the queue over time, that is, the time average of the **#n** signal.	After entity arrival and 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

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.

Parameters

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 High-Priority Customers by Sorting Entities Based on Priority"

See Also

FIFO Queue (Obsolete), LIFO Queue (Obsolete), Single Server (Obsolete)

Introduced before R2006a

Read Timer (Obsolete)

Report statistical data about named timer associated with arriving entities

Library

Timing

Description

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 **et** 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.

Parameters

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

Description

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 **Open gate upon** to `Sample time hit 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 **Open gate upon** to `Trigger 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 **Open gate upon** to `Change in signal 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 **Open gate upon** to `Function call 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.

Parameters

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
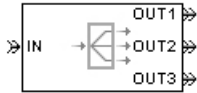
Library

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 **OUT1**, **OUT2**, **OUT3**,..., in that sequence.	The sequence of departures is always **OUT1**, **OUT2**, **OUT3**,... 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 **OUT1**, **OUT2**, **OUT3**. The second time, the copies depart in the sequence **OUT2**, **OUT3**, **OUT1**. The third time, the copies depart in the sequence **OUT3**, **OUT1**, **OUT2**. 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 **Initial seed** 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 **OUT3**, **OUT4**, **OUT1**, **OUT2**. If the random number is two on the next such occasion, then the copies depart in the sequence **OUT2**, **OUT3**, **OUT4**, **OUT1**.

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 **Number of entity output ports** parameter determines how many of these entity input 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 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.

Parameters

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

Parameters

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 **Available Resources** list. The list updates with any upstream model changes you make while the block dialog box is open.
	Add the selected resources to the **Selected Resources** table.
	Move the selected resource from the **Selected Resources** table to the **Available Resources** 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 **Selected Resources** 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 **Selected Resources** 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 **Selected Resources** table.	NA
	Move the selected resource down in order in the **Selected Resources** 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
Name	The name of the resource. Each resource must have a unique name.	Double-click the existing name, and then type the new name.
Amount Source	Whether the resource amount, that an entity requests, comes from the dialog box or an attribute.	Select `Dialog` or `Attribute`. If you select `Attribute`, 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"
Amount	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 **Amount** is 3, each entity that arrives at the Release Acquire 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 Using 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
amount	Port for arriving signals to specify the amount of the resource. You see this port only if you set **Resource amount source** to `Signal port`.

Signal Output Ports

Label	Description	Time of Update When Statistic is On
#u	Port to display the number of resources in use. You see this port only if you set **Resource amount source** to `Signal port`.	1
util	Port to display the average use of the resources. You see this port only if you select the **Average utilization, util** check box.	2

Parameters

Main Tab

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 Using Resources"

Introduced in R2015a

Resource Release (Obsolete)

Release resources that entities do not need
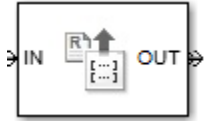
Library

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

Parameters

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 **Available Resources** list. The list updates with any upstream model changes you make while the block dialog box is open.
⤇	Add the selected resources to the **Selected Resources** table.
⬅	Move the selected resource from the **Selected Resources** table to the **Available Resources** 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 **Selected Resources** 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.

Button	Action	Notes
	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 **Selected Resources** 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 **Selected Resources** table.	N/A
	Move the selected resource down in order in the **Selected Resources** 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 Using 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

2-239

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 **Timeout interval from** to `Signal port ti`.

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.

Parameters

Schedule Timeout

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
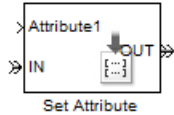
Set Attribute

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.

Parameters

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 **Available Attributes** 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 **Set Attribute** table.
	Move the selected attribute from the **Set Attribute** table to the **Available Attributes** 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 **Set Attribute** 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 **Set Attribute** table, the behavior depends on how the **Create attribute if not present** 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.

Button	Action	Notes
	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 **Set Attribute** 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
Attribute Name	The name of the attribute. Each attribute must have a unique name.	Double-click the existing name, and then type the new name.
Value From	Whether the data for the attribute value comes from the dialog box or a signal.	Select `Dialog` or `Signal port`. If you select `Signal port`, 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.
Value	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.
Vector Is 1-D	Whether the block assigns the attribute as a vector of length N or as a multidimensional array when the **Value** evaluates to an N-element 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.	Select the check box to assign the attribute as a vector of length N. Clear it to assign the attribute as a multidimensional array.

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
Attribute name	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 **Value From** to `Signal port`. The default **Name** that corresponds to each row is **Attribute*x***, where *x* = 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 and Entity Priorities"

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
wts	Signal whose updates cause write events. This signal must be an event-based signal. You see this port only if you set **Write to memory upon** to `Sample time hit from port wts`.
wtr	Trigger signal whose edges cause write events. This signal must be an event-based signal. You see this port only if you set **Write to memory upon** to `Trigger from port wtr`.
wvc	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 **Write to memory upon** to `Change in signal from port wvc`.
wfcn	Function-call signal that causes write events. This signal must be an event-based function call. You see this port only if you set **Write to memory upon** to `Function call from port wfcn`.
rts	Signal whose updates cause read events. This signal must be an event-based signal. You see this port only if you set **Read from memory upon** to `Sample time hit from port rts`.
rtr	Trigger signal whose edges cause read events. This signal must be an event-based signal. You see this port only if you set **Read from memory upon** to `Trigger from port rtr`.
rvc	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 **Read from memory upon** to `Change in signal from port rvc`.
rfcn	Function-call signal that causes read events. This signal must be an event-based function call. You see this port only if you set **Read from memory upon** to `Function call from port rfcn`.
in	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
st	0 or 1, depending on whether the block more recently processed a read or write event.	Upon write events and upon read events	1	0
mem	The value of the block's internal memory when a write event occurs.	Upon write events	1	Value of **Initial memory value** parameter
out	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.

Parameters

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

Example 2.1. 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.
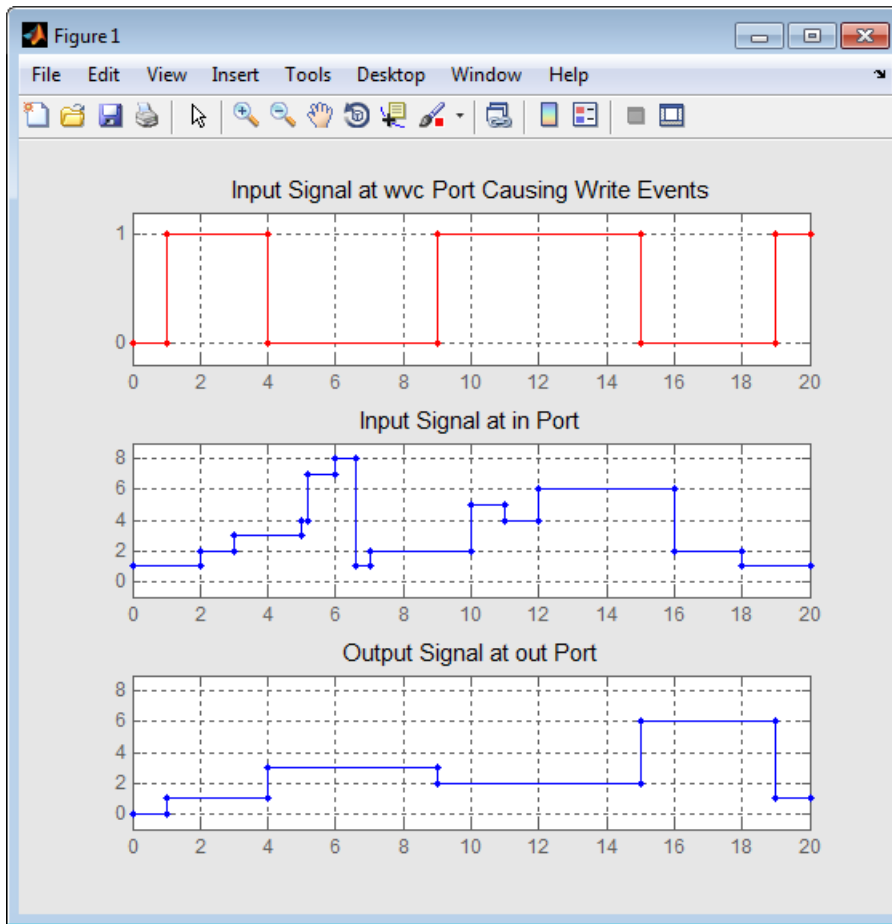
Example 2.2. Independent Read and Write Events

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.
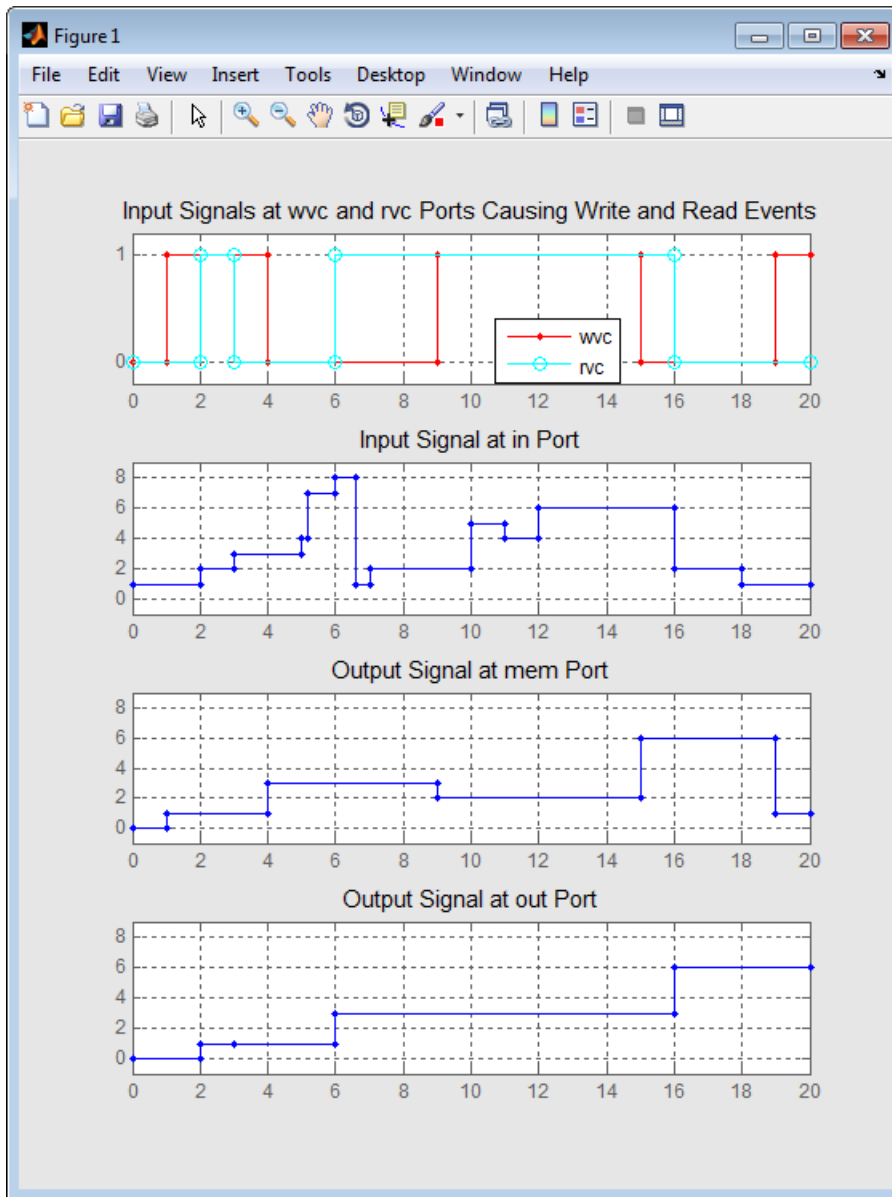
See Also

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 **in** 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 **in** 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.
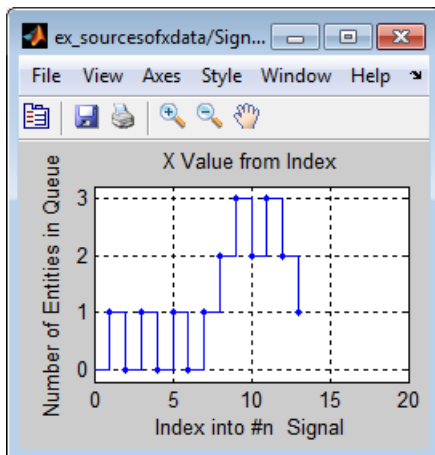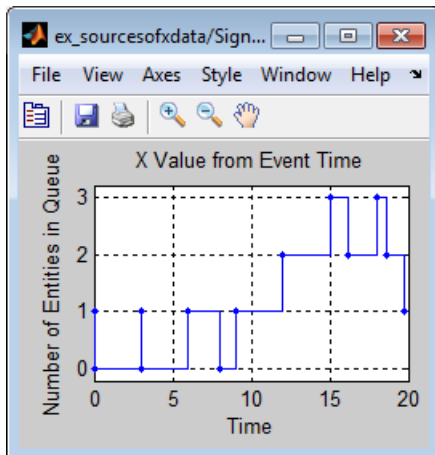
Ports

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.

Parameters

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 2-252 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

• "Create a Discrete-Event Model"

See Also

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

Introduced before R2006a

Signal-Based Function-Call Event Generator (Obsolete)

Generate function-call events in response to signal-based events

Library

Generators / EventGenerators

Description

> **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 **Generate function call only upon** to `Sample time hit from port ts`.
tr	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 **Generate function call only upon** to `Trigger from port tr`.
vc	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 **Generate function call only upon** to `Change in signal from port vc`.
fcn	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 **Generate function call only upon** to `Function call from port fcn`. Do not connect this port to an output port from the same instance of this block.

Signal Output Ports

Label	Description	Order of Update
f1	Function-call signal.	1
#f1	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.

Parameters

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

2-257

Signal-Based Function-Call Generator (Obsolete)

Convert signal-based events into function calls

Library

Generators/Function-Call Generators

Description

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 **Resolve simultaneous signal updates according to event priority**, and then set **Function-call delay from** 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 **Generate function call only upon** to Sample time hit from port ts.
tr	When this signal has a rising or falling edge, depending on the **Trigger type** parameter, the primary criterion is satisfied. This signal must be an event-based signal. You see this port only if you set **Generate function call only upon** to Trigger from port tr.
vc	When this signal increases or decreases, depending on the **Type of change in signal value** parameter, the primary criterion is satisfied. This signal must be an event-based signal. You see this port only if you set **Generate function call only upon** 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 **Generate function call only upon** 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 **f1** output port. This signal must be an event-based signal. You see this input port only if you select **Suppress function call f1 if enable signal e1 is not positive**.
e2	When this signal is 0 or negative, the block does not generate a function call at the **f2** output port. This signal must be an event-based signal. You see this input port only if you select **Suppress function call f2 if enable signal e2 is not positive**.

Signal Output Ports

Label	Description	Order of Update
f1	Function call, possibly contingent on **e1** input signal	1
f2	Function call, possibly contingent on **e2** input signal	2
#f1	Number of function calls the block has generated at the **f1** port during the simulation	3
#f2	Number of function calls the block has generated at the **f2** 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.

Parameters

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

Single Server (Obsolete)

Serve one entity for period of time

Library

Servers

Description

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 **Service time from** to `Signal port t`.

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 **Enable TO port for timed-out entities**. 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 **OUT** port since the start of the simulation.	After entity departure via the **OUT** port	4
#n	Number of entities currently in the block, either 0 or 1.	After entity arrival and after entity departure via the **OUT** or **TO** port	3
#p	Number of entities that have been preempted from this block since the start of the simulation.	After entity departure via the **P** port	4
pe	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. 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
w	Sample mean of the waiting times in this block for all entities that have departed from the **OUT** or **TO** port. An entity's waiting time might exceed its service time if the **OUT** port is blocked when the entity completes service.	After entity departure via the **OUT** or **TO** 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 On mode, updates occur after an entity departure via the **OUT** or **TO** port, and after an entity arrival. In Upon stop or pause 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 **TO** 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.

Parameters

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

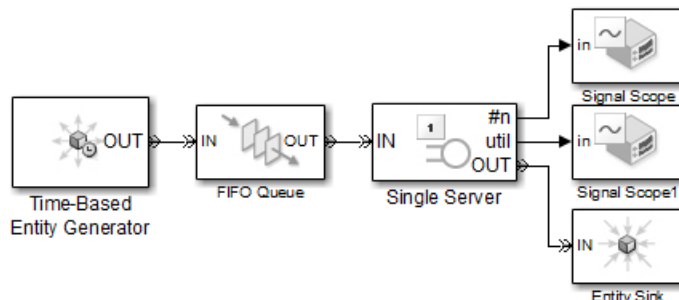Allows you to use the signal output port labeled **util**.

Number of entities timed out

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

Examples

*   "Create a Discrete-Event Model"

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.
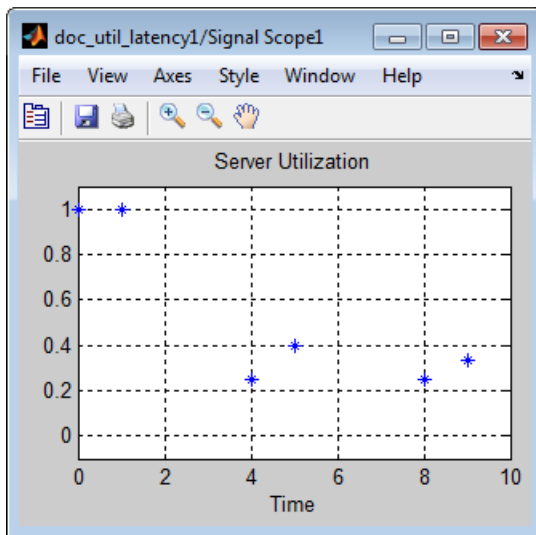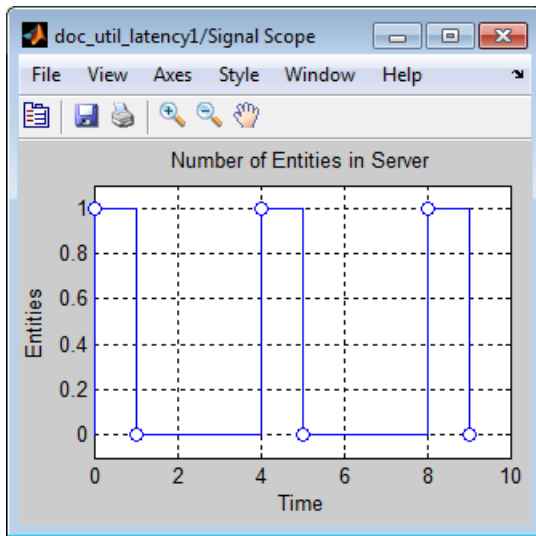
In a longer simulation, the differences in the value of **util** compared to its theoretical definition become less pronounced.
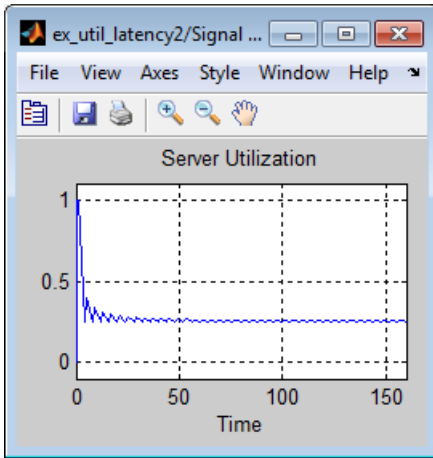
See Also

N-Server (Obsolete), Infinite Server (Obsolete)

Introduced before R2006a

Start Timer (Obsolete)

Associate named timer to each arriving entity independently and start timing

Library

Timing

Description

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.

Parameters

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

Description

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.

2-271

For an example, see "Example: Responding to Blockage" on page 2-274.

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 **Generate entity at simulation start**, 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 **Generate entities upon** to `Intergeneration time 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	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.	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.

Parameters

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 2-271.

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 2-271.

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**.
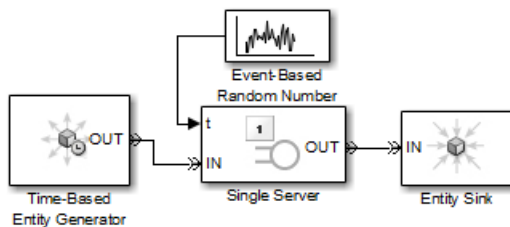
Average intergeneration time

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

Examples

- "Create 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 **OUT** 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 **OUT** 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 **OUT** 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 **OUT** 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 (Obsolete)

Introduced before R2006a

Time-Based Function-Call Generator (Obsolete)

Generate function-call events in a time-based manner.

Time-Based
Function-Call Generator
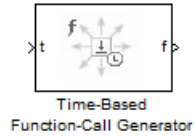
Library

Generators/Function-Call Generators

Description

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 **Period** parameter of the block dialog box
Period from port	Specified using an input signal **t** 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
t	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 **Event generation mode** parameter to `Period from port` 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 Event-Based Random Number or Event-Based Sequence 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
f	Port through which generated function-call signals depart.
#f	Number of function-call events that have been executed by the block. You only see this port if you select the **Report number of events executed, #f** 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.

Parameters

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.
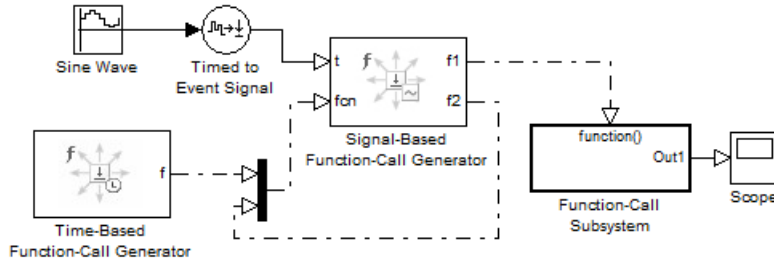
In this example, the **Event generation mode** parameter of the Time-Based Function-Call Generator block is set to `Only at simulation start`. At simulation time $T= 0$, the Time-Based Function-Call Generator block produces an initial function-call event — or seed event — to the Signal-Based Function-Call 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.
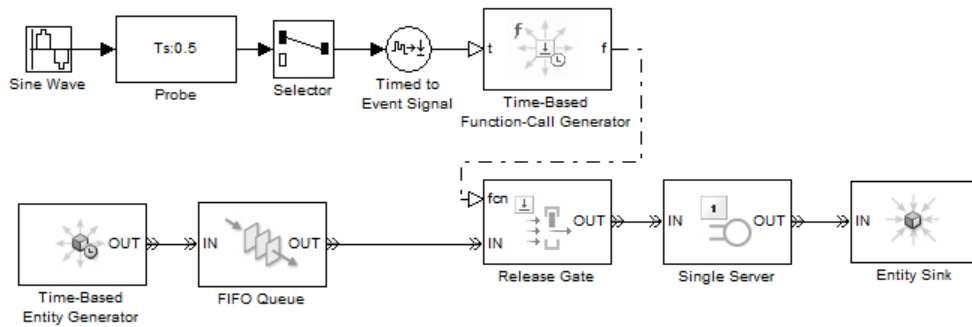
In this model, the **Event generation mode** parameter of the Time-Based Function-Call Generator block is set to `Period from port`. The Probe block detects the sample time of a sine wave signal and connects it to the input port **t** of the Time-Based Function-Call Generator block. The value at the 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.
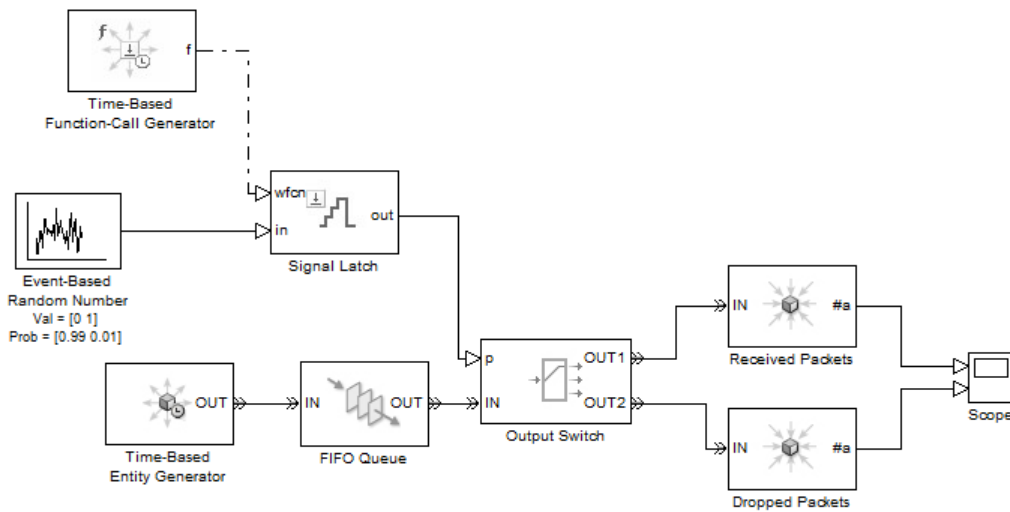
In this model, using a statistical distribution, the Event-Based Random Number block generates random values. The Time-Based Generator block periodically produces a function-call event to store the current output value of the Event-Based Random Number block in the Signal Latch block. This value is used, 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

SimEvents Sinks

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** tab 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 **Enable entity OUT port**.

Signal Output Ports

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

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

Parameters

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.

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 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 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`.

Statistics Tab

Number of points plotted

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

Examples

This section is not available for prerelease.

See Also

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

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.

Parameters

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.
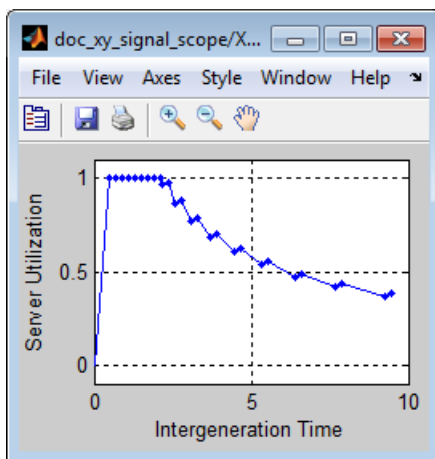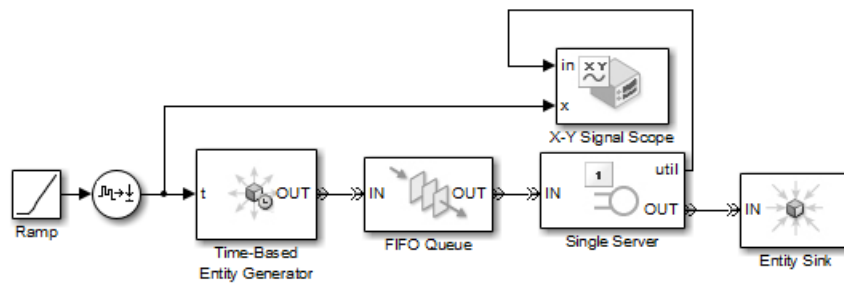
See Also

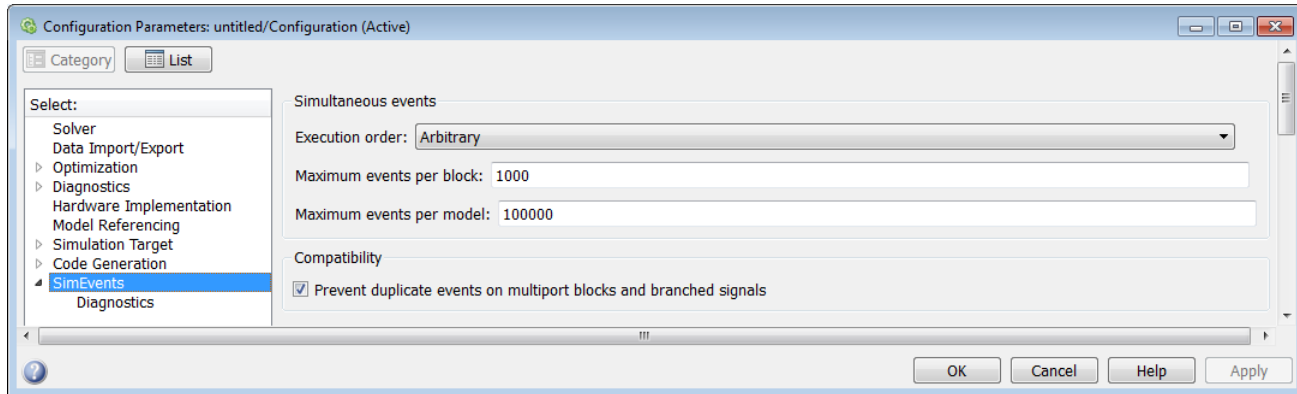Signal Scope (Obsolete), X-Y Attribute Scope (Obsolete)

Introduced before R2006a

Configuration Parameters

SimEvents Pane

Note These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.

In this section...
"SimEvents Pane Overview" on page 3-2
"Execution order" on page 3-2
"Seed for event randomization" on page 3-3
"Maximum events per block" on page 3-4
"Maximum events per model" on page 3-4
"Prevent duplicate events on multiport blocks and branched signals" on page 3-5

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`

See Also

More About

- "SimEvents Diagnostics Pane" on page 3-6

SimEvents Diagnostics Pane

> **Note** These configuration parameters are obsolete. They are available only for SimEvents releases prior to R2016a.
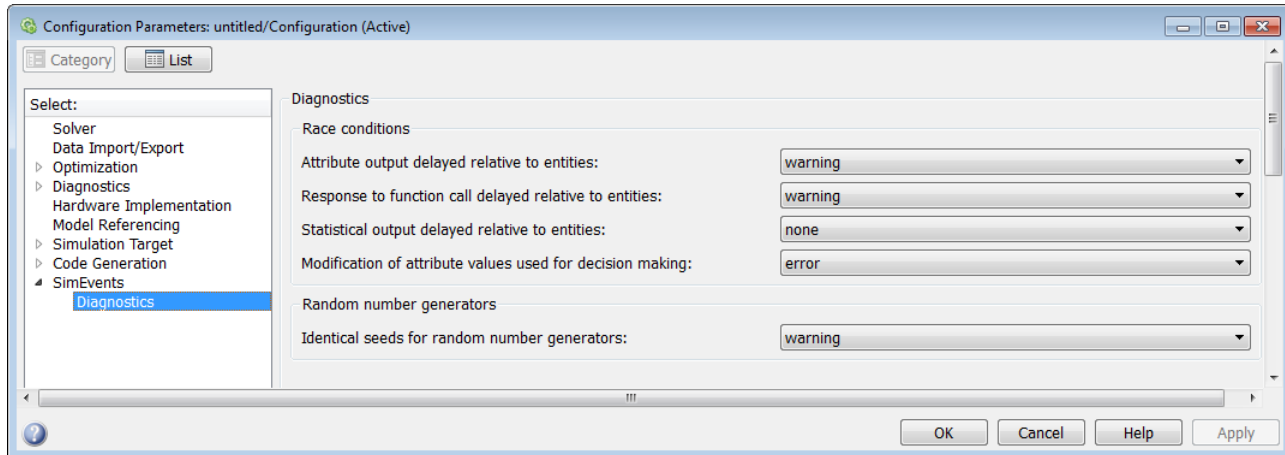
In this section...
"Diagnostics Pane Overview" on page 3-6
"Attribute output delayed relative to entities" on page 3-7
"Response to function call delayed relative to entities" on page 3-8
"Statistical output delayed relative to entities" on page 3-9
"Modification of attribute values used for decision making" on page 3-10
"Identical seeds for random number generators" on page 3-11

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 3.1. Example of Solution
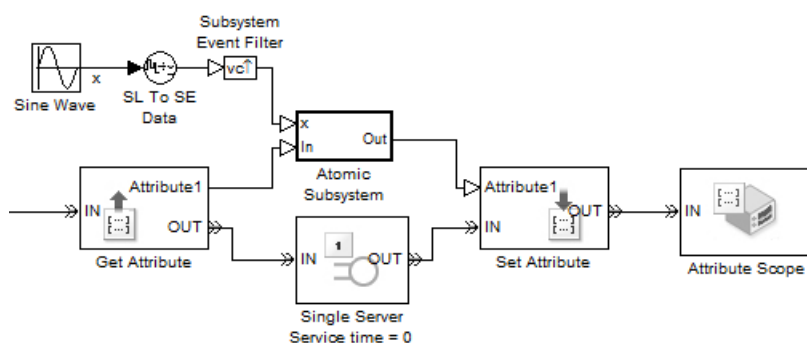
Alternatively, you might be able to address the problem by using an attribute directly instead of by using the signal output of a Get Attribute block.

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 3.2. Example of Solution
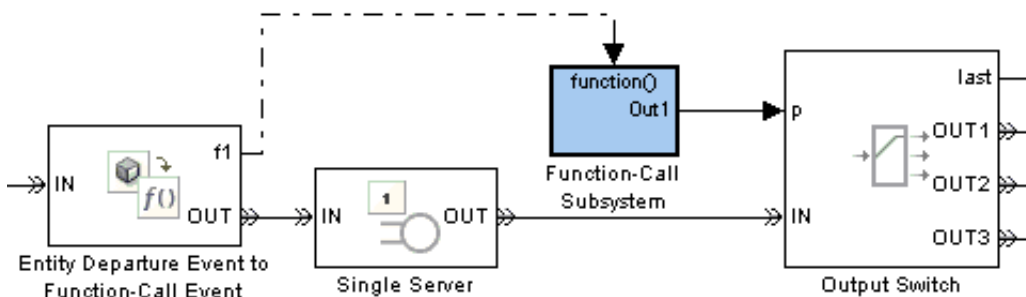
Command-Line Information
Parameter: `propDiagFcnCallOutput`
Type: double
Value: 0 | 1 | 2
Default: 2

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.
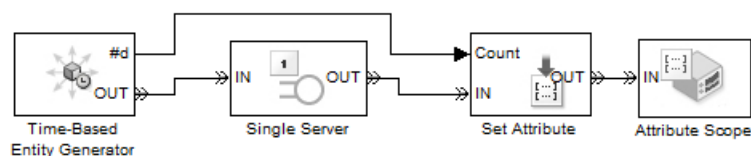
`warning`

    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 3.3. Example of Solution

Command-Line Information
Parameter: `propDiagStatOutput`

Type: double
Value: 0 | 1 | 2
Default: 1

Recommended Settings

Application	Setting
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 3.4. Example of Solution
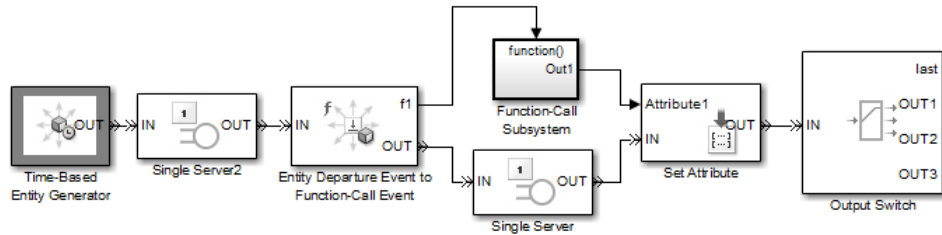
Command-Line Information
Parameter: propDiagChangeAttrib
Type: double
Value: 0 | 1 | 2
Default: 2

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

 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_randomizeseeds` 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`

See Also

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 4-2

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 >**Prevent duplicate events on multiport blocks and branched signals** check box is not selected.	In the Configuration Parameters dialog box of your model, select the SimEvents > **Prevent duplicate events on multiport blocks and branched signals** 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 4-2.

See Also

More About

*   "Consult the Upgrade Advisor"