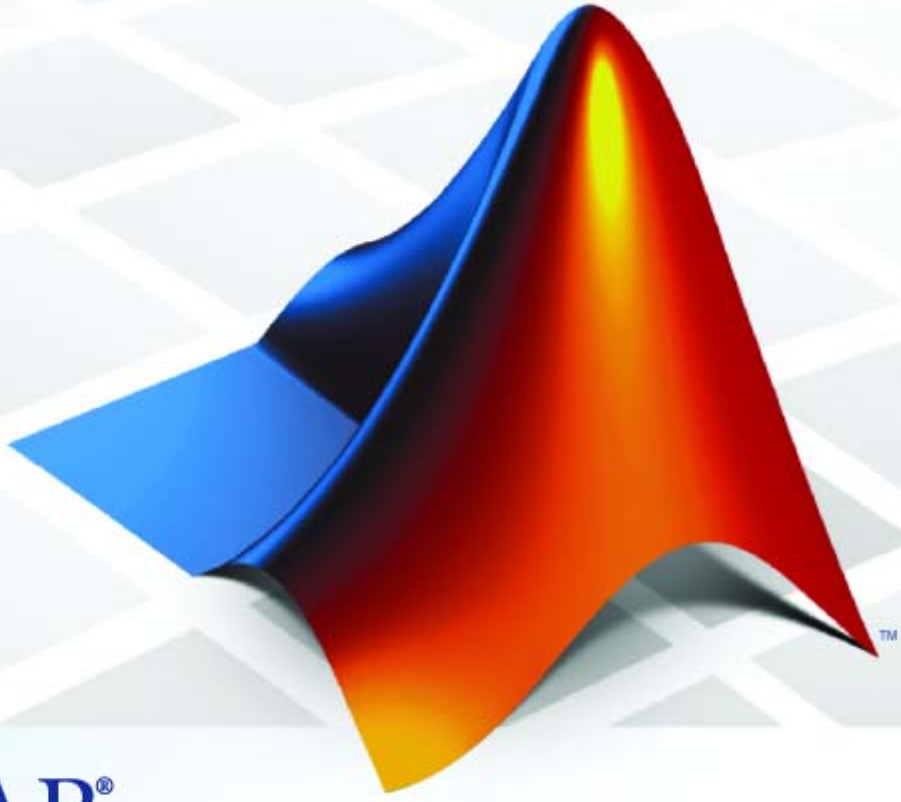
SimEvents® 2
Reference



MATLAB®
& SIMULINK®

The MathWorks™
Accelerating the pace of engineering and science

How to Contact The MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

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

For contact information about worldwide offices, see the MathWorks Web site.

SimEvents® Reference

© COPYRIGHT 2005–2009 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

The 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)

Contents

1 Function Reference

Library and Models 1-2

Simulation Settings 1-3

Debugger 1-4
- Debugger Session 1-4
- Simulation Control 1-4
- State Inspection 1-5
- Breakpoints 1-5
- Debugger Information 1-6

Seed Management 1-7

2 Alphabetical List

3 Block Reference

Generators 3-2
- Entity Generators 3-2
- Event Generators 3-2
- Signal Generators 3-2

SimEvents Sinks 3-3

Attributes 3-4

Queues **3-5**

Servers **3-6**

Routing **3-7**

Gates **3-8**

Entity Management **3-9**

Signal Management **3-10**

SimEvents Ports and Subsystems **3-11**

Timing **3-12**

Event Translation **3-13**

Probes **3-14**

SimEvents User-Defined Functions **3-15**

4 Blocks — Alphabetical List

5 Configuration Parameters

SimEvents Pane **5-2**
SimEvents Pane Overview **5-4**
Execution order **5-5**
Seed for event randomization **5-6**
Maximum events per block **5-7**

Maximum events per model 5-8

SimEvents Diagnostics Pane 5-9
Diagnostics Pane Overview 5-10
Attribute output delayed relative to entities 5-11
Response to function call delayed relative to entities 5-13
Statistical output delayed relative to entities 5-15
Modification of attribute values used for decision making 5-17
Identical seeds for random number generators 5-19

Glossary

Index

1

Function Reference

Library and Models (p. 1-2)	Open the library and example models
Simulation Settings (p. 1-3)	Configure models for discrete-event simulation
Debugger (p. 1-4)	Run and examine simulations in debug mode
Seed Management (p. 1-7)	Manage seeds of random number generators

Library and Models

simeventsdocex	Open SimEvents® documentation example model
simeventslib	Open SimEvents library

Simulation Settings

simeventsconfig	Change model settings for discrete-event simulation
simeventsstartup	Set default model settings for discrete-event simulation

Debugger

Debugger Session (p. 1-4)	Start and stop the debugger
Simulation Control (p. 1-4)	Stop, pause or resume the simulation
State Inspection (p. 1-5)	Query the model for information about the simulation
Breakpoints (p. 1-5)	Create and manage breakpoints in the debugger
Debugger Information (p. 1-6)	Start and configure the debugger

Debugger Session

sedb.quit	Quit discrete-event simulation debugging session
sedebug	Debug discrete-event simulation
se_getdbopts	SimEvents debugger options structure

Simulation Control

sedb.cont	Continue simulation until next breakpoint
sedb.runtoend	Run until end of discrete-event simulation
sedb.step	Single step in discrete-event simulation

State Inspection

sedb.blkinfo	Block information in discrete-event simulation
sedb.blklist	Blocks and their identifiers in discrete-event simulation
sedb.currentop	Current operation in discrete-event simulation
sedb.eninfo	Entity information in discrete-event simulation
sedb.evcal	Event calendar of discrete-event simulation
sedb.evinfo	Event information in discrete-event simulation
sedb.gceb	Name of currently executing block in discrete-event simulation
sedb.gcebid	Identifier of currently executing block in discrete-event simulation
sedb.gcen	Identifier of entity currently undergoing operation
sedb.gcev	Identifier of current event
sedb.simtime	Current time in discrete-event simulation

Breakpoints

sedb.bdelete	Delete breakpoints in discrete-event simulation
sedb.breakpoints	List breakpoints in discrete-event simulation
sedb.disable	Disable breakpoints in discrete-event simulation

sedb.enable	Enable breakpoints in discrete-event simulation
sedb.evbreak	Set breakpoint for execution or cancelation of event
sedb.tbreak	Set timed breakpoint in discrete-event simulation

Debugger Information

help	Display help for debugger functions
sedb.detail	Customize debugger display in discrete-event simulation

Seed Management

se_getseeds	Get seed values of random number generators in blocks
se_randomizeseeds	Randomize seeds to make them unique
se_setseeds	Set seed values for blocks with random number generators

2

Alphabetical List

help

Purpose

Display help for debugger functions

Syntax

```
help
help sedb
help functionname
```

Description

`help`, at the SimEvents debugger command prompt, displays a summary of debugger commands in the Command Window.

`help sedb` also displays a summary of debugger commands. This syntax is valid at either the SimEvents debugger command prompt or the MATLAB® command prompt.

`help functionname` displays a brief description and the syntax for `functionname` in the Command Window.

For other syntaxes that include input arguments, see `help` in the MATLAB documentation.

See Also

"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

Purpose

SimEvents debugger options structure

Syntax

```
opts_struct = se_getdbopts
```

Description

`opts_struct = se_getdbopts` returns an empty options structure that you can configure and then use as an input to the `sedebug` function. You can set the `StartFcn` field of `opts_struct` to a cell array of strings representing commands the debugger executes after the debugger session begins.

Examples

Setting Breakpoints at Startup

1. At the MATLAB command prompt, create an options structure for use in future debugger sessions:

   ```
   opts_struct = se_getdbopts;
   opts_struct.StartFcn = {'tbreak 1', 'tbreak 2'};
   ```

2. For a particular model, begin a debugger session using a syntax that includes `opts_struct` as an input argument. Using this argument causes the debugger to automatically execute the commands in the `opts_struct.StartFcn` field:

   ```
   sedebug('sedemo_outputswitch', opts_struct)
   ```

   The output in the MATLAB Command Window reflects model initialization followed by the setting of two breakpoints:

   ```
   *** SimEvents Debugger ***

   Functions | Quick Start | Debugger Help

   %==============================================================================%
   Initializing Model sedemo_outputswitch
   Set b1 : Breakpoint for first operation at or after time 1.000000
   Set b2 : Breakpoint for first operation at or after time 2.000000
   ```

3. Proceed in the simulation until the first breakpoint:

```
cont
```

The output is:

```
%=========================================================================%
Initializing Time-Based Entity Generators
    %.....................................................................%
    Scheduling EntityGeneration Event (ev1)
    : EventTime = 1.000000000000000
    : Priority  = 500
    : Block     = Time-Based Entity Generator

Hit b1 : Breakpoint for first operation at or after time 1.000000

%=========================================================================%
Executing EntityGeneration Event (ev1)              Time = 1.000000000000000
: Entity = <none>                                   Priority = 500
: Block  = Time-Based Entity Generator
```

4 To end the debugger session, at the `sedebug>>` prompt, enter this command:

```
sedb.quit
```

See Also

`sedebug`
Debugger function reference
"Debugger Efficiency Tips" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

Purpose

Get seed values of random number generators in blocks

Syntax

```
seedstruct = se_getseeds(sysid)
```

Description

`seedstruct = se_getseeds(sysid)` returns the names and seed values of all SimEvents blocks that you previously configured to use random number generators. The `sysid` input indicates the scope of the search, which is a system (including subsystems at any depth) or block.

Inputs

`sysid` — System name, subsystem path name, or block path name.

Outputs

`seedstruct` — Structure with these fields:

- `system` — Value of the `sysid` input to `se_getseeds`
- `seeds` — Structure array, of which each element has these fields:
  - `block` — Path name of a block that uses a random number generator, relative to `system`
  - `value` — Numeric seed value of the block

See Also

Seed Management Workflow for Random Number Generators demo
"Making Results Repeatable by Storing Sets of Seeds" in the SimEvents user guide documentation
"Sharing Seeds Among Models" in the SimEvents user guide documentation
`se_setseeds`
`se_randomizeseeds`

se_randomizeseeds

Purpose

Randomize seeds to make them unique

Syntax

```
se_randomizeseeds(obj)
se_randomizeseeds(obj,'Mode','All')
se_randomizeseeds(obj,'Mode','Identical',)
se_randomizeseeds(obj,'Mode','SpecifySeeds',sv)
se_randomizeseeds(...,'GlobalSeed',seed)
se_randomizeseeds(...,'Verbose',verbosity)
```

Description

`se_randomizeseeds(obj)` or `se_randomizeseeds(obj,'Mode','All')` assigns the seeds of all SimEvents blocks that use random number generators within blocks indicated by `obj`, making sure that the seeds are unique. The input `obj` is a system name, subsystem path name, block path name, or cell array of such names. If `obj` represents a system or subsystem, the function assigns seeds in subsystems of `obj` at any depth.

`se_randomizeseeds(obj,'Mode','Identical',)` changes only those seeds in the system or subsystem `obj` that appear multiple times in `obj`.

`se_randomizeseeds(obj,'Mode','SpecifySeeds',sv)` changes only those seeds that currently have a value that appears in the vector `sv`.

`se_randomizeseeds(...,'GlobalSeed',seed)` specifies a global seed that the function uses to generate seed values in a repeatable way. `seed` is a nonnegative integer. When you use this syntax, it is repeatable for a given `seed` value, assuming the underlying systems or blocks specified in `obj` do not change. However, this syntax does not guarantee uniqueness of generated seed values in order to ensure repeatability.

`se_randomizeseeds(...,'Verbose',verbosity)` explicitly reports the status of each seed change if *`verbosity`* is `'on'`.

See Also

"Running Simulations Repeatedly" in the SimEvents user guide documentation
Avoiding Identical Seeds for Random Number Generators demo
Seed Management Workflow for Random Number Generators demo

"Example: Running a Simulation and Varying a Parameter" in the SimEvents user guide documentation

`se_getseeds`

`se_setseeds`

se_setseeds

Purpose

Set seed values for blocks with random number generators

Syntax

```
se_setseeds(seedstruct)
se_setseeds(seedstruct, sysid)
oldseedstruct = se_setseeds(seedstruct...)
[oldseedstruct, status] = se_setseeds(seedstruct...)
```

Description

`se_setseeds(seedstruct)` sets the seed parameter values for all SimEvents blocks specified in `seedstruct`.

`se_setseeds(seedstruct, sysid)` applies the seed values to the system `sysid`, overriding the system specified in the `system` field of `seedstruct`.

`oldseedstruct = se_setseeds(seedstruct...)` stores the original seed values in `oldseedstruct` before setting them to the values specified in `seedstruct`.

`[oldseedstruct, status] = se_setseeds(seedstruct...)` returns status information indicating when the function does not set a seed.

Inputs

`seedstruct` — Structure having the same fields as the output of the `se_getseeds` function:

- `system` — Value of the `sysid` input to `se_getseeds`
- `seeds` — Structure array, of which each element has these fields:
  - `block` — Path name of a block that uses a random number generator, relative to `system`
  - `value` — Numeric seed value of the block

`sysid` — System name or subsystem path name.

Outputs

`oldseedstruct`	Structure representing original seed values. The form of `oldseedstruct` is the same as that of `seedstruct`.
`status`	Logical array whose nth entry is false if the nth block in `seedstruct` does not exist, is not a SimEvents block, or does not have a seed parameter in its current configuration.

See Also

"Setting Seed Values Programmatically" in the SimEvents user guide documentation
"Sharing Seeds Among Models" in the SimEvents user guide documentation
Seed Management Workflow for Random Number Generators demo
`se_getseeds`
`se_randomizeseeds`

sedb.bdelete

Purpose

Delete breakpoints in discrete-event simulation

Syntax

```
bdelete id1 id2 ...
bdelete(id_array)
bdelete all
```

Description

`bdelete id1 id2 ...` deletes the breakpoints having identifiers `id1`, `id2`, and so on. To see breakpoints and their identifiers, use `sedb.breakpoints`.

`bdelete(id_array)` deletes the breakpoints in the cell array `id_array`.

`bdelete all` deletes all breakpoints.

Examples

Setting, Manipulating, and Using Timed Breakpoints

This example illustrates the syntax of breakpoint commands by creating and manipulating some breakpoints.

1. Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

   ```
   sedebug('sedemo_timeout')
   ```

2. Establish and view breakpoints by entering these commands at the `sedebug>>` prompt:

   ```
   tbreak 0.5
   tbreak 1
   tbreak 1.5
   breakpoints
   ```

   The output lists the breakpoints:

   ```
   Set b1 : Breakpoint for first operation at or after time 0.500000

   Set b2 : Breakpoint for first operation at or after time 1.000000

   Set b3 : Breakpoint for first operation at or after time 1.500000
   ```

```
List of Breakpoints:

 ID      Type      Value     Enabled
 b1      Timed     0.5       yes
 b2      Timed     1         yes
 b3      Timed     1.5       yes
```

3 Delete, disable, and enable some of the breakpoints. View the resulting list:

```
bdelete b2
disable b3
disable b1
enable b3
breakpoints
```

The output is:

```
Deleted b2 : Breakpoint for first operation at or after time 1.000000

Disabled b3 : Breakpoint for first operation at or after time 1.500000

Disabled b1 : Breakpoint for first operation at or after time 0.500000

Enabled b3 : Breakpoint for first operation at or after time 1.500000


List of Breakpoints:

 ID      Type      Value     Enabled
 b1      Timed     0.5       no
 b3      Timed     1.5       yes
```

4 Run the simulation until the first enabled breakpoint:

```
cont
```

The partial output shows that `b3` is the breakpoint at which the debugger stops. The debugger ignores the disabled breakpoint, `b1`, and does not consider the previously deleted breakpoint, `b2`.

```
Hit b3 : Breakpoint for first operation at or after time 1.500000


%=============================================================================%
Executing EntityGeneration Event (ev7)                Time = 1.500000000000000
: Entity = <none>                                     Priority = 1
: Block  = Time-Based Entity Generator
```

5 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.breakpoints`
`sedb.disable`
`sedb.enable`
"Using Breakpoints During Debugging" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

Purpose

Block information in discrete-event simulation

Syntax

```
blkinfo(blkid)
blkinfo(blkname)
blk_struct = blkinfo(...)
```

Description

`blkinfo(blkid)` displays information about the block with identifier `blkid`. Use `sedb.blklist` to obtain a list of blocks and their identifiers.

`blkinfo(blkname)` displays information about the block with path name `blkname`.

`blk_struct = blkinfo(...)` returns a structure that stores information about the block.

Inputs

`blkid`	Block identifier of a SimEvents library block other than the blocks listed as exceptions.
`blkname`	Path name of a SimEvents library block other than the blocks listed as exceptions.

Exceptions: Blocks That Do Not Provide Information

- Conn
- Discrete Event Inport
- Discrete Event Outport
- Discrete Event Subsystem
- Discrete Event Signal to Workspace
- Event-Based Random Number
- Event-Based Sequence
- Initial Value
- Instantaneous Event Counting Scope

- Subsystem Configuration

Outputs

`blk_struct`	Structure that stores information about the block. The `blk_struct` fields and the blocks for which each field appears are described in the next table.

Fields of Output Structure

Field	Block	Description
`Time`	All blocks	Current simulation time
`Block`	All blocks	Path name of the block
`BlockID`	All blocks	Block identifier
`BlockType`	All blocks	Type of block
`Capacity`	All queue and server blocks	Number of entities that the block can store at any one time
`SelectedInputPort`	Input Switch	Index of selected entity input port
`SelectedOutputPort`	Output Switch	Index of selected entity output port
`GateStatus`	Enabled Gate	`'open'` or `'closed'`
`InputEntity`	Replicate	Identifier of the entity being replicated
`CurrentReplica`	Replicate	Identifier of the copy of the input entity that has just been created and is about to advance
`Status`	Replicate, Entity Combiner, and Entity Splitter	Status of block. For possible values, see Block Status Values in blk_struct.Status on page 2-16.
`MemoryValue`	Signal Latch	Value of the block's internal memory

Fields of Output Structure (Continued)

Field	Block	Description
`Count`	Entity Departure Counter	Number of entities that have departed from this block and arrived at subsequent storage blocks
`Entities`	All blocks that possess an entity input port	Structure array, of which each element has these fields: • `ID` — Entity identifier • `Status` (queue and server blocks only) — Status of the entity with respect to the block. For possible values, see Entity Status Values in blk_struct.Entities.Status on page 2-16. • `Event` (server blocks only) — Identifier of the service completion event for the entity • `EventTime` (server blocks only) — Time of the service completion event for the entity • `TimeInQueue` (queue blocks only) — Length of time the entity has been in the queue

Block Status Values in blk_struct.Status

Block	Possible Status Values
Replicate	`'Inactive'` `'Replicating X/Y'`. *X* is the index of the entity output port through which the replica will depart, and *Y* is the **Number of entity output ports** parameter of the block
Entity Combiner	`'Inactive'` `'Combining'`
Entity Splitter	`'Inactive'` `'Splitting'`

Entity Status Values in blk_struct.Entities.Status

Block	Possible Status Values

All queue blocks	`'Advancing'`
	`'Queuing'`
	`'Queued'`
	`'Timed Out'`

All server blocks	`'Advancing'`
	`'In Service'`
	`'Service Completed'`
	`'Preempted'`
	`'Timed Out'`

Examples

Viewing Block Information

1 Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

```
sedebug('sedemo_start_timer_read_timer')
```

2 Proceed in the simulation by entering these commands at the `sedebug>>` prompt:

```
tbreak 2.51
cont
```

The partial output indicates that the server is about to execute a service completion event:

```
Hit b1 : Breakpoint for first operation at or after time 2.510000

%=============================================================================%
Executing ServiceCompletion Event (ev8)              Time = 2.581301297500028
: Entity = en4                                       Priority = 1
: Block  = Infinite Server
```

3 View information about the block that is about to execute the event:

```
blkname = gceb
blkid = gcebid
```

The output is:

```
blkname =

sedemo_start_timer_read_timer/Infinite Server


blkid =

blk7
```

4 Proceed further in the simulation:

```
step
```

The output shows that the entity is about to depart from the server:

```
%.....................................................................%
    Entity Advancing (en4)
    : From = Infinite Server
    : To   = Read Timer
```

5 Use the identifier, `blkid`, to get information about the block and the status of entities in it:

```
% Display information in Command Window.
blkinfo(blkid)
% Store information in structure.
blkdetails = blkinfo(blkid)
% Store status of entities in cell array.
blkentities = {blkdetails.Entities.Status}'
```

The output is:

```
Infinite Server Current State                          T = 2.581301297500028
Block (blk7): Infinite Server

Entities (Capacity = Inf):
 Pos     ID        Status                  Event     EventTime
 1       en2       In Service              ev4       2.7303849396254689
 2       en4       Advancing               ev8       2.581301297500028
 3       en5       In Service              ev10      2.974736350905304


blkdetails =

         Time: 2.5813
        Block: 'sedemo_start_timer_read_timer/Infinite Server'
      BlockID: 'blk7'
    BlockType: 'Infinite Server'
     Capacity: Inf
     Entities: [1x3 struct]


blkentities =
```

```
    'In Service'
    'Advancing'
    'In Service'
```

6 Get more information about one of the entities by using data from the `blkdetails` structure as an input argument to the `eninfo` function:

```
adv_eninfo = eninfo(blkdetails.Entities(1).ID);
time_in_system = adv_eninfo.Timers.ElapsedTime
```

The output is:

```
time_in_system =

    1.5813
```

7 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.blklist`
`sedb.gceb`
`sedb.eninfo`
"Inspecting Entities, Blocks, and Events" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

sedb.blklist

Purpose

Blocks and their identifiers in discrete-event simulation

Syntax

```
blklist
blk_cell = blklist
```

Description

`blklist` displays a list of event-based blocks and block identifiers in the model being debugged. The list includes all blocks for which `blkinfo` is valid. The list excludes virtual subsystems and Discrete Event Subsystem blocks, but includes relevant blocks inside such subsystems. You can click hyperlinks in the Command Window to highlight the blocks in the model window.

`blk_cell = blklist` returns the same information as `blklist` in a cell array of strings. The first column of `blk_cell` contains block identifiers and the corresponding cells in the second column contain block path names.

Examples

Viewing the Block List and a Variable

1. Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

   ```
   sedebug('sedemo_count_attributes')
   ```

2. View the list of blocks by entering this command at the `sedebug>>` prompt:

   ```
   blklist
   ```

   The output is:

   ```
   List of blocks in model: sedemo_count_attributes

      blk1                Entity Index
      blk2                Time-Based Entity Generator
      blk3                Set Attribute
      blk4                Get Attribute
      blk5                Entity Sink
   ```

3 Store the list of blocks in a variable and examine one row of the cell array:

```
x = blklist;
x{1,:}
```

The output indicates the identifier and path name of one block in the model:

```
ans =

blk1


ans =

sedemo_count_attributes/Entity Index
```

4 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.blkinfo`

"Inspecting Blocks" in the SimEvents user guide documentation

"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

sedb.breakpoints

Purpose

List breakpoints in discrete-event simulation

Syntax

```
breakpoints
b_struct = breakpoints
```

Description

`breakpoints` displays a list of all breakpoints in the simulation. The list includes disabled breakpoints, as well as breakpoints that the debugger already hit or might never hit.

`b_struct = breakpoints` returns a structure array that stores information about breakpoints in the simulation. Each structure in the array has the fields that are described in the table.

Field	Description
`ID`	Breakpoint identifier
`Type`	Type of breakpoint
`Value`	Value associated with the breakpoint, as a string
`Enabled`	1 if the breakpoint is enabled; 0 otherwise

Examples

Viewing Breakpoints

1. Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

   ```
   sedebug('sedemo_timeout')
   ```

2. Establish breakpoints and then view them in the Command Window, by entering these commands at the `sedebug>>` prompt:

   ```
   tbreak 0.5
   tbreak 1
   tbreak 1.5
   breakpoints
   ```

The output confirms the setting of breakpoints and displays the list of breakpoints:

```
Set b1 : Breakpoint for first operation at or after time 0.500000

Set b2 : Breakpoint for first operation at or after time 1.000000

Set b3 : Breakpoint for first operation at or after time 1.500000


List of Breakpoints:

 ID        Type        Value       Enabled
 b1        Timed       0.5         yes
 b2        Timed       1           yes
 b3        Timed       1.5         yes
```

3 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

Manipulating the Array of Breakpoints

To manipulate the structure array that is the output from `breakpoints`:

1 Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

```
sedebug('sedemo_timeout')
```

2 Establish breakpoints and then record them in a structure variable, by entering these commands at the `sedebug>>` prompt:

```
tbreak 0.5
tbreak 1
tbreak 1.5
b = breakpoints
```

The output confirms the setting of breakpoints and gives an initial view of the structure `b`:

```
Set b1 : Breakpoint for first operation at or after time 0.500000

Set b2 : Breakpoint for first operation at or after time 1.000000

Set b3 : Breakpoint for first operation at or after time 1.500000


b =

1x3 struct array with fields:
    ID
    Type
    Value
    Enabled
```

3 View information about the first breakpoint:

```
b1 = b(1)
```

The output is a structure:

```
b1 =

         ID: 'b1'
       Type: 'Timed'
      Value: '0.5'
    Enabled: 1
```

4 Store the IDs of the enabled breakpoints in a cell array, `bid`:

```
idx = find([b.Enabled]);
bid = {b(idx).ID}
```

The output is:

```
bid =
```

```
    'b1'    'b2'    'b3'
```

5 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.tbreak`
`sedb.evbreak`
`sedb.bdelete`
`sedb.disable`
"Defining a Breakpoint" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

Purpose

Continue simulation until next breakpoint

Syntax

```
cont
```

Description

`cont` continues the simulation until it reaches the next breakpoint or the end, whichever comes first.

Examples

1 Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

```
sedebug('sedemo_timeout')
```

2 Proceed in the simulation, by entering these commands at the `sedebug>>` prompt:

```
tbreak 2
detail none
cont
```

The output ends with a message describing the context of the simulation at T=2:

```
Set b1 : Breakpoint for first operation at or after time 2.000000


  Event Operations  (ev)   :  off
  Entity Operations (en)   :  off
  Event Calendar    (cal)  :  off


Hit b1 : Breakpoint for first operation at or after time 2.000000

%=============================================================================%
Executing EntityGeneration Event (ev10)              Time = 2.000000000000000
: Entity = <none>                                    Priority = 1
: Block  = Time-Based Entity Generator
```

3 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.runtoend`

`sedb.step`

"Defining a Breakpoint" in the SimEvents user guide documentation

"Using Breakpoints During Debugging" in the SimEvents user guide documentation

"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

sedb.currentop

Purpose

Current operation in discrete-event simulation

Syntax

```
currentop
```

Description

`currentop` displays the most recent message in the simulation log. If this message represents a dependent operation, the output also includes a message about the most recent independent operation in the simulation log.

Definitions

An *independent operation* is one of these:

- Initialization of the model or any Time-Based Entity Generator blocks in the model. For more information, see "Initialization Messages".
- Execution of an event on the event calendar. However, if an event is not scheduled on the event calendar, it cannot be the basis of an independent operation. To learn which events are on the event calendar, see "Role of the Event Calendar".
- Detection by a reactive port of a relevant update in a time-based input signal. You can think of these relevant updates as zero crossings or level crossings. However, if the input signal is event-based or if the input port is not a reactive port, the update is not an independent operation.

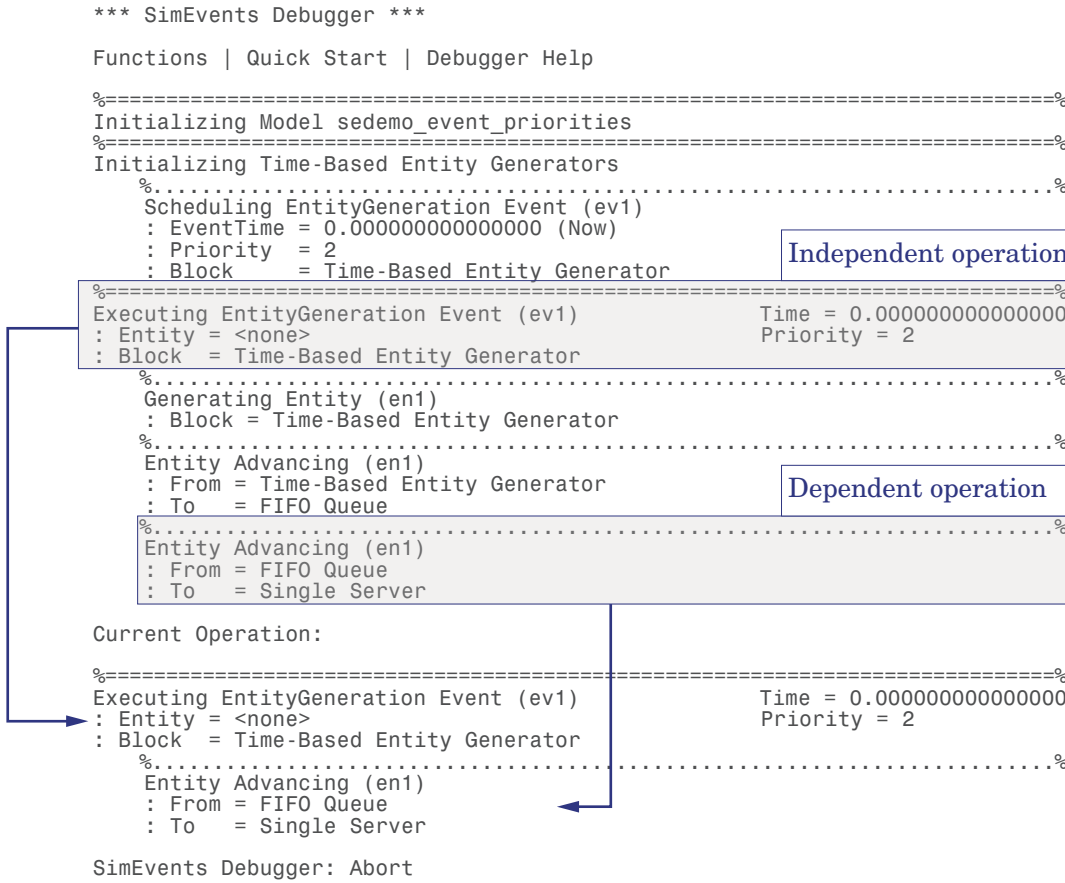
A *dependent operation* is a consequence of an independent operation.

For more information, see "Independent Operations and Consequences in the Debugger" in the SimEvents user guide documentation.

Examples

Comparing Current Operation Display with Simulation Log

The sample output shows that the display of the current operation includes the last entry in the simulation log, which is the current operation. The display also includes the last unindented entry in the simulation log, representing the latest independent operation.



To reproduce this result, at the MATLAB command prompt, enter:

```
opts_struct = se_getdbopts;
opts_struct.StartFcn={'step over','step',...
   'step','step','currentop','quit'};
sedebug('sedemo_event_priorities',opts_struct)
```

See Also

`sedb.simtime`

"Simulation Log in the Debugger" in the SimEvents user guide documentation

"Independent Operations and Consequences in the Debugger" in the SimEvents user guide documentation

"Inspecting the Current Point in the Debugger" in the SimEvents user guide documentation

"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

Purpose

Customize debugger display in discrete-event simulation

Syntax

```
detail(param1, value1, param2, value2,...)
detail('none')
detail('default')
detail('all')
prev = detail(...)
detail(s)
detail
s = detail
```

Description

`detail(param1, value1, param2, value2,...)` configures the debugger to show or omit certain kinds of messages in the simulation log. You can specify one, two, or three pairs of parameter names, *`paramN`*, and parameter values, *`valueN`*. Valid parameter names and parameter values are described in the tables in "Inputs" on page 2-32.

`detail('none')` configures the debugger to omit all simulation log messages, except upon reaching breakpoints. This syntax is the same as `detail('en',0,'ev',0,'cal',0)`. You cannot use the `step` function to step to anything other than breakpoints when the debugger omits all simulation log messages.

`detail('default')` resets the detail settings to their default values. This syntax is the same as `detail('en',1,'ev',1,'cal',0)`.

`detail('all')` configures the debugger to show all simulation log messages. This syntax is the same as `detail('en',1,'ev',1,'cal',1)`.

`prev = detail(...)` configures the debugger based on the inputs, and also returns a structure that describes the previous detail settings.

`detail(s)` uses the structure `s` to establish detail settings.

`detail` displays the current detail settings.

`s = detail` returns a structure that describes the current detail settings.

Inputs

param1, *param2*, and so on	Names of detail settings. Valid names are in Names of Detail Settings on page 2-32.
value1, *value2*, and so on	Values of detail settings. Valid values are in Values of Detail Settings on page 2-32.
`s`	Structure having three fields describing the detail settings that you want. Field names and values are the same as the names and values in the following tables.

Names of Detail Settings

Parameter Name	Description
`'ev'`	Event operations, except that independent operations representing event executions appear in the simulation log if at least one detail setting is 1, even if the `'ev'` setting is 0.
`'en'`	Entity operations
`'cal'`	Event calendar information

Values of Detail Settings

Parameter Value	Description
0	Omit messages, except upon reaching a breakpoint. Omitting entity messages or event messages also means you cannot use the `step` function to step to an operation that corresponds to an omitted message, unless a breakpoint coincides with the operation.
1	Show messages

Outputs

`prev`	Structure that describes the previous detail settings before changing them. Field names and values are the same as the names and values in Names of Detail Settings on page 2-32 and Values of Detail Settings on page 2-32.
`s`	Structure that describes the current detail settings. Field names and values are the same as the names and values in Names of Detail Settings on page 2-32 and Values of Detail Settings on page 2-32.

Examples

Configuring Displays for Breakpoints and Stepping

This example suppresses the simulation log until the debugger reaches a breakpoint and then enables the simulation log to make subsequent `step` operations more informative.

1 Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

```
sedebug('sedemo_timeout')
```

2 Move forward in the simulation. At the `sedebug>>` prompt,enter:

```
tbreak 2
prev = detail('none')
cont
```

The output ends with the context of the simulation at T=2:

```
Set b1 : Breakpoint for first operation at or after time 2.000000

prev =

    ev: 1
```

```
     en: 1
    cal: 0


Hit b1 : Breakpoint for first operation at or after time 2.000000

%=============================================================================%
Executing EntityGeneration Event (ev10)              Time = 2.000000000000000
: Entity = <none>                                    Priority = 1
: Block  = Time-Based Entity Generator
```

3 Now that the simulation is at a point of interest, configure the debugger to show the simulation log. Then start to move forward in the simulation:

```
detail(prev)
step over
```

The output confirms the change in detail settings and then shows the result of enabling the simulation log:

```
Event Operations  (ev)   :  on
Entity Operations (en)   :  on
Event Calendar    (cal)  :  off


  %.....................................................................%
  Generating Entity (en4)
  : Block = Time-Based Entity Generator
  %.....................................................................%
  Entity Advancing (en4)
  : From = Time-Based Entity Generator
  : To   = Schedule Timeout
  %.....................................................................%
  Scheduling Timeout Event (ev11)
  : EventTime = 3.000000000000000
  : Priority  = 1700
  : Entity    = en4
```

```
    : Block     = Schedule Timeout
    %.....................................................................%
    Entity Advancing (en4)
    : From = Schedule Timeout
    : To   = Infinite Server
    %.....................................................................%
    Scheduling ServiceCompletion Event (ev12)
    : EventTime = 2.581301297500028
    : Priority  = 1
    : Entity    = en4
    : Block     = Infinite Server
    %.....................................................................%
    Scheduling EntityGeneration Event (ev13)
    : EventTime = 2.500000000000000
    : Priority  = 1
    : Block     = Time-Based Entity Generator
%=============================================================================%
Executing Timeout Event (ev5)                       Time = 2.000000000000000
: Entity = en2                                      Priority = 1700
: Block  = Infinite Server
```

4 To end the debugger session, at the `sedebug>>` prompt, enter:

```
sedb.quit
```

See Also

`sedb.cont`
`sedb.tbreak`
`sedb.evbreak`
"Customizing the Debugger Simulation Log" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

sedb.disable

Purpose

Disable breakpoints in discrete-event simulation

Syntax

```
disable id1 id2 ...
disable(id_array)
disable all
```

Description

`disable id1 id2 ...` disables the breakpoints having identifiers `id1`, `id2`, and so on. To see breakpoints and their identifiers, use `sedb.breakpoints`. A disabled breakpoint remains in the list but the `sedb.cont` function ignores it.

`disable(id_array)` disables the breakpoints in the cell array `id_array`.

`disable all` disables all breakpoints.

Examples

Setting, Manipulating, and Using Timed Breakpoints

This example illustrates the syntax of breakpoint commands by creating and manipulating some breakpoints.

1. Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

   ```
   sedebug('sedemo_timeout')
   ```

2. Establish and view breakpoints by entering these commands at the `sedebug>>` prompt:

   ```
   tbreak 0.5
   tbreak 1
   tbreak 1.5
   breakpoints
   ```

   The output lists the breakpoints:

   ```
   Set b1 : Breakpoint for first operation at or after time 0.500000

   Set b2 : Breakpoint for first operation at or after time 1.000000
   ```

```
Set b3 : Breakpoint for first operation at or after time 1.500000


List of Breakpoints:

 ID      Type      Value     Enabled
 b1      Timed     0.5       yes
 b2      Timed     1         yes
 b3      Timed     1.5       yes
```

3 Delete, disable, and enable some of the breakpoints. View the resulting list:

```
bdelete b2
disable b3
disable b1
enable b3
breakpoints
```

The output is:

```
Deleted b2 : Breakpoint for first operation at or after time 1.000000

Disabled b3 : Breakpoint for first operation at or after time 1.500000

Disabled b1 : Breakpoint for first operation at or after time 0.500000

Enabled b3 : Breakpoint for first operation at or after time 1.500000


List of Breakpoints:

 ID      Type      Value     Enabled
 b1      Timed     0.5       no
 b3      Timed     1.5       yes
```

4 Run the simulation until the first enabled breakpoint:

```
cont
```

The partial output shows that `b3` is the breakpoint at which the debugger stops. The debugger ignores the disabled breakpoint, `b1`, and does not consider the previously deleted breakpoint, `b2`.

```
Hit b3 : Breakpoint for first operation at or after time 1.500000

%=============================================================================%
Executing EntityGeneration Event (ev7)                Time = 1.500000000000000
: Entity = <none>                                     Priority = 1
: Block  = Time-Based Entity Generator
```

5 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.breakpoints`
`sedb.enable`
`sedb.bdelete`
"Using Breakpoints During Debugging" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

Purpose

Enable breakpoints in discrete-event simulation

Syntax

```
enable id1 id2 ...
enable(id_array)
enable all
```

Description

`enable id1 id2 ...` enables the breakpoints having identifiers `id1`, `id2`, and so on. To see breakpoints and their identifiers, use `sedb.breakpoints`.

`enable(id_array)` enables the breakpoints in the cell array, `id_array`.

`enable all` enables all breakpoints.

Examples

Setting, Manipulating, and Using Timed Breakpoints

This example illustrates the syntax of breakpoint commands by creating and manipulating some breakpoints.

1. Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

   ```
   sedebug('sedemo_timeout')
   ```

2. Establish and view breakpoints by entering these commands at the `sedebug>>` prompt:

   ```
   tbreak 0.5
   tbreak 1
   tbreak 1.5
   breakpoints
   ```

   The output lists the breakpoints:

   ```
   Set b1 : Breakpoint for first operation at or after time 0.500000

   Set b2 : Breakpoint for first operation at or after time 1.000000

   Set b3 : Breakpoint for first operation at or after time 1.500000
   ```

```
List of Breakpoints:

 ID      Type      Value     Enabled
 b1      Timed     0.5       yes
 b2      Timed     1         yes
 b3      Timed     1.5       yes
```

3 Delete, disable, and enable some of the breakpoints. View the resulting list:

```
bdelete b2
disable b3
disable b1
enable b3
breakpoints
```

The output is:

```
Deleted b2 : Breakpoint for first operation at or after time 1.000000

Disabled b3 : Breakpoint for first operation at or after time 1.500000

Disabled b1 : Breakpoint for first operation at or after time 0.500000

Enabled b3 : Breakpoint for first operation at or after time 1.500000


List of Breakpoints:

 ID      Type      Value     Enabled
 b1      Timed     0.5       no
 b3      Timed     1.5       yes
```

4 Run the simulation until the first enabled breakpoint:

```
cont
```

The partial output shows that `b3` is the breakpoint at which the debugger stops. The debugger ignores the disabled breakpoint, `b1`, and does not consider the previously deleted breakpoint, `b2`.

```
Hit b3 : Breakpoint for first operation at or after time 1.500000

%=============================================================================%
Executing EntityGeneration Event (ev7)                Time = 1.500000000000000
: Entity = <none>                                     Priority = 1
: Block  = Time-Based Entity Generator
```

5 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.breakpoints`
`sedb.disable`
`sedb.bdelete`
"Using Breakpoints During Debugging" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

sedb.eninfo

Purpose

Entity information in discrete-event simulation

Syntax

```
eninfo(enid)
en_struct = eninfo(enid)
```

Description

`eninfo(enid)` displays information about the location, attributes, timers, and timeouts on the entity with identifier `enid`. To find the identifier of an entity, see "Obtaining Entity Identifiers" in the SimEvents user guide documentation.

`en_struct = eninfo(enid)` returns a structure that stores information about the entity. The following table describes the `en_struct` fields.

Field	Description
`Time`	Current simulation time
`Location`	Path name of the block containing the entity
`Attributes`	Structure whose field names and values match those of the attributes of the entity
`Timers`	Structure array, of which each element has these fields: • `Tag` — Timer tag • `ElapsedTime` — Time elapsed since the timer started
`Timeouts`	Structure array, of which each element has these fields: • `Tag` — Timeout tag • `TimeOfTimeout` — Scheduled time of timeout event • `Event`— Identifier of timeout event

Examples

Viewing Entity Information

1 Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

```
sedebug('sedemo_star_routing')
```

2 Proceed in the simulation by entering these commands at the `sedebug>>` prompt:

```
tbreak 0.1
cont
```

The partial output shows the entity identifier, `en1`, in the display:

```
Hit b1 : Breakpoint for first operation at or after time 0.100000

%=============================================================================%
Executing ServiceCompletion Event (ev2)              Time = 0.300000000000000
: Entity = en1                                       Priority = 5
: Block  = Distribution Center/Infinite Server
```

3 Get the ID of an entity:

```
enid = gcen % Get ID of current entity.
```

The workspace variable, `enid`, holds the same entity identifier, `en1`, from the display:

```
enid =

en1
```

4 Use the identifier, `enid`, to get information about the entity, by entering these commands at the `sedebug>>` prompt:

```
% Display information in Command Window.
eninfo(enid)
% Store information in structure.
```

```
endetails = eninfo(enid)
% View attributes.
enattrs = endetails.Attributes
% View one attribute.
enServiceProcess = enattrs.ServiceProcess
% Equivalently, view one attribute starting from endetails.
enServiceProcess = endetails.Attributes.ServiceProcess;
```

The output is:

```
Entity (en1) Current State                          T = 0.300000000000000
Location: Distribution Center/Infinite Server

Attributes:
  Name                Value
  CurrentRoute        1
  CurrentServiceTime  2
  CurrentStep         2
  JobClass            1
  JobID               1
  JobServiceStatus    [1x15]
  LastServiceLocation 0
  ServiceProcess      [6x1]
  ServiceTime         [6x1]

Timeouts, Timers: None


endetails =

         Time: 0.3
     Location: 'sedemo_star_routing/Distribution Center/Infinite Server'
   Attributes: [1x1 struct]
       Timers: [0x0 struct]
     Timeouts: [0x0 struct]
```

```
enattrs =

           CurrentRoute: 1
     CurrentServiceTime: 2
            CurrentStep: 2
               JobClass: 1
                  JobID: 1
        JobServiceStatus: [0 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN]
    LastServiceLocation: 0
         ServiceProcess: [6x1 double]
            ServiceTime: [6x1 double]


enServiceProcess =

     1
     2
     4
     2
     3
     5
```

5 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.gcen`
`sedb.blkinfo`
`sedb.evinfo`
`sedb.currentop`
"Inspecting Entities, Blocks, and Events" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

sedb.evbreak

Purpose

Set breakpoint for execution or cancelation of event

Syntax

```
evbreak(evid)
bid = evbreak(evid)
```

Description

`evbreak(evid)` sets a breakpoint for execution or cancelation of the event with identifier, `evid`, in the simulation. Use `sedb.evcal` to obtain a list of events on the event calendar and their identifiers.

`bid = evbreak(evid)` returns the identifier of the breakpoint.

Examples

Setting and Using an Event Breakpoint

Set a breakpoint on a service completion event that occurs at T=1. Because that event is not the first event the simulation executes at that time, the example shows how an event breakpoint differs from a timed breakpoint.

1. Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

   ```
   sedebug('sedemo_server_service_time')
   ```

2. Proceed by entering this command at the `sedebug>>` prompt:

   ```
   step over
   ```

   The output is:

   ```
   %=============================================================================%
   Initializing Time-Based Entity Generators
       %.....................................................................%
       Scheduling EntityGeneration Event (ev1)
       : EventTime = 0.000000000000000 (Now)
       : Priority  = 500
       : Block     = Time-Based Entity Generator
       %.....................................................................%
       Scheduling EntityGeneration Event (ev2)
       : EventTime = 0.000000000000000 (Now)
   ```

```
    : Priority  = 500
    : Block     = Time-Based Entity Generator 1
    %.....................................................................%
    Scheduling EntityGeneration Event (ev3)
    : EventTime = 0.000000000000000 (Now)
    : Priority  = 500
    : Block     = Time-Based Entity Generator 3
  %=========================================================================%
  Executing EntityGeneration Event (ev1)               Time = 0.000000000000000
  : Entity = <none>                                    Priority = 500
  : Block  = Time-Based Entity Generator
```

3 Proceed further:

```
step over
```

The output, with bold added, shows that a service completion event with identifier, `ev4`, appears on the event calendar. The scheduled time of the event is T=1.

```
    %.....................................................................%
    Generating Entity (en1)
    : Block = Time-Based Entity Generator
    %.....................................................................%
    Entity Advancing (en1)
    : From = Time-Based Entity Generator
    : To   = Single Server
    %.....................................................................%
    Scheduling ServiceCompletion Event (ev4)
    : EventTime = 1.000000000000000
    : Priority  = 500
    : Entity    = en1
    : Block     = Single Server
    %.....................................................................%
    Scheduling EntityGeneration Event (ev5)
    : EventTime = 1.000000000000000
    : Priority  = 500
    : Block     = Time-Based Entity Generator
```

```
%=============================================================================%
Executing EntityGeneration Event (ev2)               Time = 0.000000000000000
: Entity = <none>                                    Priority = 500
: Block  = Time-Based Entity Generator 1
```

4 Set a breakpoint that causes the debugger to stop when it is about to execute the service completion event, `ev4`:

```
evbreak ev4
```

The output confirms the operation:

```
Set b1 : Breakpoint for execution or cancelation of event ev4
```

5 Run the simulation until the breakpoint:

```
cont
```

The partial output shows that the debugger proceeds through a different event at T=1, and stops upon hitting the breakpoint at event `ev4`. A timed breakpoint at T=1 would have caused the debugger to stop upon the first event at this time, `ev6`.

```
%=============================================================================%
Executing ServiceCompletion Event (ev6)              Time = 1.000000000000000
: Entity = en2                                       Priority = 300
: Block  = Infinite Server
    %.....................................................................%
    Entity Advancing (en2)
    : From = Infinite Server
    : To   = Entity Sink2
    %.....................................................................%
    Destroying Entity (en2)
    : Block = Entity Sink2

Hit b1 : Breakpoint for execution or cancelation of event ev4

%=============================================================================%
Executing ServiceCompletion Event (ev4)              Time = 1.000000000000000
```

```
: Entity = en1                                          Priority = 500
: Block  = Single Server
```

6 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.cont`
`sedb.evcal`
`sedb.breakpoints`
`sedb.bdelete`
"Defining a Breakpoint" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

sedb.evcal

Purpose

Event calendar of discrete-event simulation

Syntax

```
evcal
cal_struct = evcal
```

Description

`evcal` displays the list of events on the event calendar. The event in progress or the event that the application selected for execution, whichever one applies, appears with notation => to the left of the event identifier. The display includes this event unless all its immediate consequences are complete.

`cal_struct = evcal` returns a structure that stores information about the event calendar. The `cal_struct` fields are described in the next table.

Field	Description
`Time`	Current simulation time
`ExecutingEvent`	Structure that describes the event in progress or selected for execution.The structure has these fields: • `ID` — Event identifier • `EventType` — Type of event • `EventTime` — Scheduled time of event • `Priority` — Priority of event • `Entity` — Identifier of entity associated with the event • `Block` — Path name of block that executes the event
`PendingEvents`	Structure array that describes events that are not in progress or selected for execution. Each structure in the array has these fields: • `ID` — Event identifier • `EventType` — Type of event • `EventTime` — Scheduled time of event • `Priority` — Priority of event • `Entity` — Identifier of entity associated with the event • `Block` — Path name of block that executes the event

Examples

Viewing the Event Calendar

1 Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

```
sedebug('sedemo_timeout')
```

2 Proceed in the simulation, by entering these commands at the `sedebug>>` prompt:

```
tbreak 2
detail none
cont
```

The output ends with a message describing the context of the simulation at T=2:

```
Set b1 : Breakpoint for first operation at or after time 2.000000


  Event Operations  (ev)   :  off
  Entity Operations (en)   :  off
  Event Calendar    (cal)  :  off


Hit b1 : Breakpoint for first operation at or after time 2.000000

%=============================================================================%
Executing EntityGeneration Event (ev10)              Time = 2.000000000000000
: Entity = <none>                                    Priority = 1
: Block  = Time-Based Entity Generator
```

3 View the event calendar:

```
evcal % View event calendar.
```

The following output lists all events on the event calendar:

```
Current Time = 2.000000000000000
%-------------------------------------------------------------------------------%
Events in the Event Calendar
    ID    EventTime          EventType          Priority Entity   Block
 => ev10  2.000000000000000  EntityGeneration   1        <none>   Time-Based Entity Generator
    ev5   2.000000000000000  Timeout            1700     en2      Infinite Server
    ev6   2.730384939625469  ServiceCompletion  1        en2      Infinite Server
```

4 Store the event calendar as a structure and get more information about the first pending event:

```
cal_struct = evcal % Store event calendar as a structure.
ev5struct = cal_struct.PendingEvents(1)
```

The output is below:

```
cal_struct =

             Time: 2
    ExecutingEvent: [1x1 struct]
     PendingEvents: [2x1 struct]


ev5struct =

           ID: 'ev5'
    EventType: 'Timeout'
    EventTime: 2
     Priority: '1700'
       Entity: 'en2'
        Block: 'sedemo_timeout/Infinite Server'
```

5 Find the types of pending events whose scheduled time is the current simulation time:

```
idx = find([cal_struct.PendingEvents.EventTime] == simtime);
simult_event_types = {cal_struct.PendingEvents(idx).EventType}'
```

The output is:

```
simult_event_types =

    'Timeout'
```

6 Store all event times in a vector:

```
ev_times = cal_struct.ExecutingEvent.EventTime;
ev_times = [ev_times, cal_struct.PendingEvents.EventTime]'
```

The output is:

```
ev_times =

    2.0000
    2.0000
    2.7304
```

7 To end the debugger session, at the `sedebug>>` prompt, enter:

```
sedb.quit
```

After the debugger session ends, `cal_struct` and the other variables remain in the workspace.

See Also

`sedb.evinfo`
`sedb.simtime`
`sedb.currentop`
"Viewing the Event Calendar" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

Purpose

Event information in discrete-event simulation

Syntax

```
ev_struct = evinfo(evid)
```

Description

`ev_struct = evinfo(evid)` returns a structure that stores information about the event with identifier `evid`. Use `sedb.evcal` to obtain a list of events on the event calendar and their identifiers. The following table describes the `ev_struct` fields.

Field	Description
`ID`	Event identifier
`EventType`	Type of event
`EventTime`	Scheduled time of event
`Priority`	Priority of event
`Entity`	Identifier of entity associated with the event
`Block`	Path name of block that executes the event

Examples

Viewing Event Information

1 Begin a debugger session for a particular model. At the MATLAB command prompt, enter:

```
sedebug('sedemo_count_attributes')
```

2 Proceed in the simulation. At the `sedebug>>` prompt, enter:

```
step
```

The output shows the identifier, `ev1`, in the display:

```
%=============================================================================%
Initializing Time-Based Entity Generators
    %.....................................................................%
    Scheduling EntityGeneration Event (ev1)
```

```
: EventTime = 0.000000000000000 (Now)
: Priority  = 1
: Block     = Time-Based Entity Generator
```

3 View information about the event being scheduled:

```
evid = gcev
evdetails = evinfo(evid)
```

The output shows the same identifier, `ev1`, in the workspace variable, `evid`, and the `ID` field of the structure, `evdetails`:

```
evid =

ev1


evdetails =

          ID: 'ev1'
   EventType: 'EntityGeneration'
   EventTime: 0
    Priority: '1'
      Entity: ''
       Block: [1x51 char]
```

4 End the debugger session. At the `sedebug>>` prompt, enter:

```
sedb.quit
```

See Also

`sedb.evcal`
`sedb.gcev`
"Inspecting Entities, Blocks, and Events" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

Purpose

Name of currently executing block in discrete-event simulation

Syntax

```
blkname = gceb
```

Description

`blkname = gceb` returns the path name of the block associated with the current operation. If the current operation is not associated with a block, `blkname` is empty. If an entity is advancing, `blkname` indicates the block from which the entity departs.

Examples

Viewing Block Information

1 Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

```
sedebug('sedemo_start_timer_read_timer')
```

2 Proceed in the simulation by entering these commands at the `sedebug>>` prompt:

```
tbreak 2.51
cont
```

The partial output indicates that the server is about to execute a service completion event:

```
Hit b1 : Breakpoint for first operation at or after time 2.510000

%=============================================================================%
Executing ServiceCompletion Event (ev8)              Time = 2.581301297500028
: Entity = en4                                       Priority = 1
: Block  = Infinite Server
```

3 View information about the block that is about to execute the event:

```
blkname = gceb
blkid = gcebid
```

The output is:

```
blkname =

sedemo_start_timer_read_timer/Infinite Server


blkid =

blk7
```

4 Proceed further in the simulation:

```
step
```

The output shows that the entity is about to depart from the server:

```
%.....................................................................%
    Entity Advancing (en4)
    : From = Infinite Server
    : To   = Read Timer
```

5 Use the identifier, `blkid`, to get information about the block and the status of entities in it:

```
% Display information in Command Window.
blkinfo(blkid)
% Store information in structure.
blkdetails = blkinfo(blkid)
% Store status of entities in cell array.
blkentities = {blkdetails.Entities.Status}'
```

The output is:

```
Infinite Server Current State                          T = 2.581301297500028
Block (blk7): Infinite Server

Entities (Capacity = Inf):
 Pos     ID        Status                  Event     EventTime
 1       en2       In Service              ev4       2.7303849396254689
 2       en4       Advancing               ev8       2.581301297500028
```

```
 3       en5       In Service               ev10      2.974736350905304


blkdetails =

         Time: 2.5813
        Block: 'sedemo_start_timer_read_timer/Infinite Server'
      BlockID: 'blk7'
    BlockType: 'Infinite Server'
     Capacity: Inf
     Entities: [1x3 struct]


blkentities =

    'In Service'
    'Advancing'
    'In Service'
```

6 Get more information about one of the entities by using data from the `blkdetails` structure as an input argument to the `eninfo` function:

```
adv_eninfo = eninfo(blkdetails.Entities(1).ID);
time_in_system = adv_eninfo.Timers.ElapsedTime
```

The output is:

```
time_in_system =

    1.5813
```

7 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.blkinfo`
`sedb.gcebid`

`sedb.gcev`
`sedb.gcen`
`sedb.currentop`
“Inspecting the Current Point in the Debugger” in the SimEvents user guide documentation
“Overview of the SimEvents Debugger” in the SimEvents user guide documentation

Purpose

Identifier of currently executing block in discrete-event simulation

Syntax

```
blkid = gcebid
```

Description

`blkid = gcebid` returns the identifier of the block associated with the current operation. If the current operation is not associated with a block, `blkid` is empty. If an entity is advancing, `blkid` indicates the block from which the entity departs.

Examples

Viewing Block Information

1 Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

```
sedebug('sedemo_start_timer_read_timer')
```

2 Proceed in the simulation by entering these commands at the `sedebug>>` prompt:

```
tbreak 2.51
cont
```

The partial output indicates that the server is about to execute a service completion event:

```
Hit b1 : Breakpoint for first operation at or after time 2.510000

%=============================================================================%
Executing ServiceCompletion Event (ev8)              Time = 2.581301297500028
: Entity = en4                                       Priority = 1
: Block  = Infinite Server
```

3 View information about the block that is about to execute the event:

```
blkname = gceb
blkid = gcebid
```

The output is:

```
blkname =

sedemo_start_timer_read_timer/Infinite Server


blkid =

blk7
```

4 Proceed further in the simulation:

```
step
```

The output shows that the entity is about to depart from the server:

```
%.....................................................................%
    Entity Advancing (en4)
    : From = Infinite Server
    : To   = Read Timer
```

5 Use the identifier, `blkid`, to get information about the block and the status of entities in it:

```
% Display information in Command Window.
blkinfo(blkid)
% Store information in structure.
blkdetails = blkinfo(blkid)
% Store status of entities in cell array.
blkentities = {blkdetails.Entities.Status}'
```

The output is:

```
Infinite Server Current State                          T = 2.581301297500028
Block (blk7): Infinite Server

Entities (Capacity = Inf):
 Pos     ID        Status                  Event     EventTime
 1       en2       In Service              ev4       2.7303849396254689
 2       en4       Advancing               ev8       2.581301297500028
```

```
 3       en5       In Service               ev10      2.974736350905304


blkdetails =

         Time: 2.5813
        Block: 'sedemo_start_timer_read_timer/Infinite Server'
      BlockID: 'blk7'
    BlockType: 'Infinite Server'
     Capacity: Inf
     Entities: [1x3 struct]


blkentities =

    'In Service'
    'Advancing'
    'In Service'
```

6 Get more information about one of the entities by using data from the `blkdetails` structure as an input argument to the `eninfo` function:

```
adv_eninfo = eninfo(blkdetails.Entities(1).ID);
time_in_system = adv_eninfo.Timers.ElapsedTime
```

The output is:

```
time_in_system =

    1.5813
```

7 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.blkinfo`
`sedb.gceb`

`sedb.gcev`
`sedb.gcen`
`sedb.currentop`
“Inspecting the Current Point in the Debugger” in the SimEvents user guide documentation
“Overview of the SimEvents Debugger” in the SimEvents user guide documentation

Purpose

Identifier of entity currently undergoing operation

Syntax

```
enid = gcen
```

Description

`enid = gcen` returns the identifier of the entity that undergoes the current operation. If the current operation does not apply to a unique entity, `enid` is empty.

Examples

Viewing Entity Information

1 Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

```
sedebug('sedemo_star_routing')
```

2 Proceed in the simulation by entering these commands at the `sedebug>>` prompt:

```
tbreak 0.1
cont
```

The partial output shows the entity identifier, `en1`, in the display:

```
Hit b1 : Breakpoint for first operation at or after time 0.100000

%=============================================================================%
Executing ServiceCompletion Event (ev2)              Time = 0.300000000000000
: Entity = en1                                       Priority = 5
: Block  = Distribution Center/Infinite Server
```

3 Get the ID of an entity:

```
enid = gcen % Get ID of current entity.
```

The workspace variable, `enid`, holds the same entity identifier, `en1`, from the display:

```
enid =
```

```
en1
```

4 Use the identifier, `enid`, to get information about the entity, by entering these commands at the `sedebug>>` prompt:

```
% Display information in Command Window.
eninfo(enid)
% Store information in structure.
endetails = eninfo(enid)
% View attributes.
enattrs = endetails.Attributes
% View one attribute.
enServiceProcess = enattrs.ServiceProcess
% Equivalently, view one attribute starting from endetails.
enServiceProcess = endetails.Attributes.ServiceProcess;
```

The output is:

```
Entity (en1) Current State                          T = 0.300000000000000
Location: Distribution Center/Infinite Server

Attributes:
  Name                Value
  CurrentRoute        1
  CurrentServiceTime  2
  CurrentStep         2
  JobClass            1
  JobID               1
  JobServiceStatus    [1x15]
  LastServiceLocation 0
  ServiceProcess      [6x1]
  ServiceTime         [6x1]

Timeouts, Timers: None


endetails =
```

```
          Time: 0.3
      Location: 'sedemo_star_routing/Distribution Center/Infinite Server'
    Attributes: [1x1 struct]
        Timers: [0x0 struct]
      Timeouts: [0x0 struct]


enattrs =

          CurrentRoute: 1
     CurrentServiceTime: 2
           CurrentStep: 2
              JobClass: 1
                 JobID: 1
       JobServiceStatus: [0 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN]
    LastServiceLocation: 0
         ServiceProcess: [6x1 double]
            ServiceTime: [6x1 double]


enServiceProcess =

     1
     2
     4
     2
     3
     5
```

5 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.eninfo`
`sedb.gcev`

`sedb.gceb`

`sedb.currentop`

“Inspecting the Current Point in the Debugger” in the SimEvents user guide documentation

“Overview of the SimEvents Debugger” in the SimEvents user guide documentation

Purpose

Identifier of current event

Syntax

```
evid = gcev
```

Description

`evid = gcev` returns the identifier of the event associated with the current operation, or an empty string, if the current operation does not change the event calendar. A change to the event calendar can be the scheduling, execution, or cancelation of an event.

Examples

Event Associated with the Current Operation

1. Begin a debugger session for a particular model. At the MATLAB command prompt, enter:

   ```
   sedebug('sedemo_async_stateflow_ent')
   ```

2. Proceed in the simulation. At the `sedebug>>` prompt, enter:

   ```
   tbreak 1
   cont
   ```

   The output ends with the current operation, which is the execution of the event with identifier `ev6`:

   ```
   %=============================================================================%
   Executing ServiceCompletion Event (ev6)              Time = 1.000000000000000
   : Entity = en2                                       Priority = 500
   : Block  = Asynchronous Execution/Single Server
   ```

3. View information about the event associated with the current operation. This event is the service completion event being scheduled, not the entity generation event whose execution causes the scheduling of the service completion event.

   ```
   evid = gcev
   evdetails = evinfo(evid)
   ```

The output shows the same identifier, `ev6`, in the workspace variable, `evid`, and the `ID` field of the structure, `evdetails`:

```
evid =

ev6


evdetails =

           ID: 'ev6'
    EventType: 'ServiceCompletion'
    EventTime: 1
     Priority: '500'
       Entity: 'en2'
        Block: 'sedemo_async_stateflow_ent/Asynchronous Execution/Single Server'
```

4 Proceed further in the simulation:

```
step
```

The output is:

```
%.....................................................................%
Scheduling EntityRequest Event (ev7)
: EventTime = 1.000000000000000 (Now)
: Priority  = SYS2
: Block     = Asynchronous Execution/Single Server
```

5 View information about the event associated with the current operation. This event is the entity request event being scheduled, not the service completion event whose execution causes the scheduling of the entity request event.

```
evid_next = gcev
evdetails_next = evinfo(evid_next)
```

The output refers to the event identifier, `ev7`:

```
evid_next =

ev7


evdetails_next =

           ID: 'ev7'
    EventType: 'EntityRequest'
    EventTime: 1
     Priority: 'SYS2'
       Entity: ''
        Block: 'sedemo_async_stateflow_ent/Asynchronous Execution/Single Server'
```

6 End the debugger session. At the `sedebug>>` prompt, enter:

```
sedb.quit
```

See Also

`sedb.evinfo`
`sedb.evcal`
`sedb.gcen`
`sedb.gceb`
`sedb.currentop`
"Inspecting Entities, Blocks, and Events" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

sedb.quit

Purpose

Quit discrete-event simulation debugging session

Syntax

```
quit
sedb.quit
```

Description

`quit`, at the SimEvents debugger command prompt, stops the simulation and exits the debugging session.

`sedb.quit` is the same as the preceding `quit` syntax. Specifying the `sedb` package prevents you from inadvertently ending the MATLAB session if you enter the command at the incorrect command prompt.

See Also

`sedb.runtoend`

"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

Purpose

Run until end of discrete-event simulation

Syntax

```
runtoend
```

Description

`runtoend` continues the simulation until the end, ignoring timed and event breakpoints. At the end of the simulation, the debugging session ends.

See Also

`sedb.quit`
`sedb.cont`
"Stopping the Debugger" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

sedb.simtime

Purpose

Current time in discrete-event simulation

Syntax

```
t = simtime
```

Description

`t = simtime`, at the SimEvents debugger command prompt, returns the current simulation time.

Examples

Setting a Breakpoint Relative to the Current Time

This example advances the simulation by two seconds using a timed breakpoint.

1. At the MATLAB command prompt, start the debugger on an example model:

   ```
   sedebug('sedemo_count_attributes');
   ```

2. At the debugger prompt, use the current time to set a breakpoint:

   ```
   t = simtime;
   tbreak(t+2)
   ```

   The following output confirms the setting of the breakpoint:

   ```
   Set b1 : Breakpoint for first operation at or after time 2.000000
   ```

3. Run the simulation until the breakpoint:

   ```
   cont
   ```

   The output describes simulation behavior and then confirms that the debugger has reached the breakpoint. Here is an excerpt:

   ```
   Hit b1 : Breakpoint for first operation at or after time 2.000000

   %=============================================================================%
   Executing EntityGeneration Event (ev3)                Time = 2.000000000000000
   : Entity = <none>                                     Priority = 1
   : Block  = Time-Based Entity Generator
   ```

4 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.currentop`

"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

Purpose

Single step in discrete-event simulation

Syntax

```
step
step in
step out
step over
```

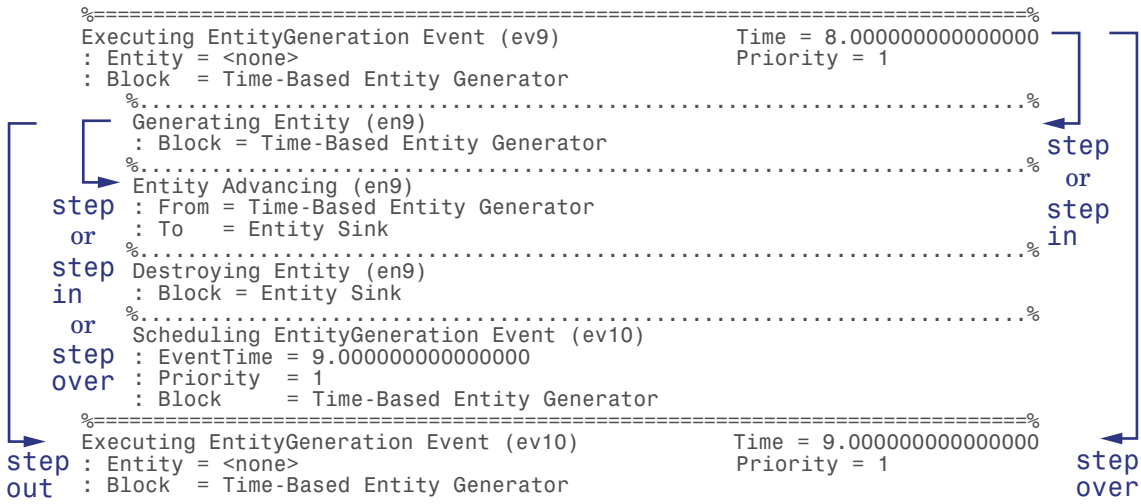
Description

`step` or `step in`, at the SimEvents debugger command prompt, advances the simulation by the smallest possible step in the debugger.

`step out` advances the simulation to the next independent operation that appears in the simulation log.

`step over` advances the simulation to the next message at the same level in the simulation log.

The figure illustrates commands for navigating the two-level hierarchy of the simulation log.



See Also

`sedb.runtoend`

"Confirming Event-Based Behavior Using the SimEvents Debugger" in the SimEvents getting started documentation

"Stepping Through the Simulation" in the SimEvents user guide documentation

"Independent Operations and Consequences in the Debugger" in the SimEvents user guide documentation

"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

sedb.tbreak

Purpose

Set timed breakpoint in discrete-event simulation

Syntax

```
tbreak(t)
tbreak t
tid = tbreak(t)
```

Description

`tbreak(t)` or `tbreak t`, at the SimEvents debugger command prompt, sets a breakpoint for time `t`. If you subsequently enter `cont`, the simulation stops at the first operation at or after time `t`. The argument, `t`, is a numeric scalar or a string that represents a number.

`tid = tbreak(t)` returns the identifier of the breakpoint.

Examples

Setting, Manipulating, and Using Timed Breakpoints

This example illustrates the syntax of breakpoint commands by creating and manipulating some breakpoints.

1. Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

   ```
   sedebug('sedemo_timeout')
   ```

2. Establish and view breakpoints by entering these commands at the `sedebug>>` prompt:

   ```
   tbreak 0.5
   tbreak 1
   tbreak 1.5
   breakpoints
   ```

   The output lists the breakpoints:

   ```
   Set b1 : Breakpoint for first operation at or after time 0.500000

   Set b2 : Breakpoint for first operation at or after time 1.000000

   Set b3 : Breakpoint for first operation at or after time 1.500000
   ```

```
List of Breakpoints:

 ID      Type      Value     Enabled
 b1      Timed     0.5       yes
 b2      Timed     1         yes
 b3      Timed     1.5       yes
```

3 Delete, disable, and enable some of the breakpoints. View the resulting list:

```
bdelete b2
disable b3
disable b1
enable b3
breakpoints
```

The output is:

```
Deleted b2 : Breakpoint for first operation at or after time 1.000000

Disabled b3 : Breakpoint for first operation at or after time 1.500000

Disabled b1 : Breakpoint for first operation at or after time 0.500000

Enabled b3 : Breakpoint for first operation at or after time 1.500000


List of Breakpoints:

 ID      Type      Value     Enabled
 b1      Timed     0.5       no
 b3      Timed     1.5       yes
```

4 Run the simulation until the first enabled breakpoint:

```
cont
```

The partial output shows that `b3` is the breakpoint at which the debugger stops. The debugger ignores the disabled breakpoint, `b1`, and does not consider the previously deleted breakpoint, `b2`.

```
Hit b3 : Breakpoint for first operation at or after time 1.500000


%=============================================================================%
Executing EntityGeneration Event (ev7)                Time = 1.500000000000000
: Entity = <none>                                     Priority = 1
: Block  = Time-Based Entity Generator
```

5 To end the debugger session, enter this command at the `sedebug>>` prompt:

```
sedb.quit
```

See Also

`sedb.cont`
`sedb.breakpoints`
`sedb.bdelete`
"Defining a Breakpoint" in the SimEvents user guide documentation
"Overview of the SimEvents Debugger" in the SimEvents user guide documentation

Purpose

Debug discrete-event simulation

Syntax

```
sedebug(sys)
sedebug(sys, opts_struct)
```

Description

`sedebug(sys)` begins a SimEvents debugger session on a model. `sys` is the name of the top-level system as a string. Enter this command at the MATLAB command prompt. After the prompt changes to `sedebug>>`, you can enter debugging commands.

`sedebug(sys, opts_struct)` applies debugging options in `opts_struct`. Use `se_getdbopts` to obtain an options structure in the correct format. In particular, you can set the `StartFcn` field of `opts_struct` to a cell array of strings representing commands the debugger executes after the debugger session begins.

Examples

Starting and Ending a Debugger Session

1. Begin a debugger session for a particular model by entering this command at the MATLAB command prompt:

   ```
   sedebug('sedemo_outputswitch')
   ```

   The output in the MATLAB Command Window is:

   ```
   *** SimEvents Debugger ***

   Functions | Quick Start | Debugger Help

   %=============================================================================%
   Initializing Model sedemo_outputswitch
   sedebug>>
   ```

2. To end the debugger session, enter this command at the `sedebug>>` prompt:

   ```
   sedb.quit
   ```

See Also

`se_getdbopts`
`sedb.quit`
Debugger function reference
“Overview of the SimEvents Debugger” in the SimEvents user guide documentation
“Debugger Efficiency Tips” in the SimEvents user guide documentation

Purpose

Change model settings for discrete-event simulation

Syntax

```
simeventsconfig
simeventsconfig(sys,'des')
simeventsconfig(sys,'hybrid')
```

Description

`simeventsconfig` changes the current system settings to values appropriate for discrete-event simulation (DES) modeling.

`simeventsconfig(sys,'des')` changes settings of the specified system to values appropriate for discrete-event simulation modeling. In particular, this syntax changes the solver to a variable-step discrete solver.

`simeventsconfig(sys,'hybrid')` changes settings of the specified system to values appropriate for modeling systems that combine time-driven and discrete-event-driven behavior. In particular, this syntax changes the solver to a continuous solver, `ode45`.

Algorithm

This function assigns the following simulation settings.

Parameter	Setting for DES Systems	Setting for Hybrid Systems
SolverName	VariableStepDiscrete	ode45
SolverType	Variable-step	Variable-step
MaxStep	inf	inf
SaveTime	off	off
SaveOutput	off	off
AlgebraicLoopMsg	error	error
SolverPrmCheckMsg	none	none

For information about the settings in the table, see “Model Parameters” in the Simulink® documentation. For information about discrete and continuous solvers, see “Solvers” in the Simulink documentation.

See Also

`simeventsstartup`

Purpose

Open SimEvents documentation example model

Syntax

```
simeventsdocex(mname)
```

Description

`simeventsdocex(mname)` opens the SimEvents documentation example model `mname`, modifying the MATLAB path if necessary.

simeventslib

Purpose

Open SimEvents library

Syntax

```
simeventslib
```

Description

`simeventslib` opens the main SimEvents library.

Purpose

Set default model settings for discrete-event simulation

Syntax

```
simeventsstartup('des')
simeventsstartup('hybrid')
```

Description

`simeventsstartup('des')` changes the default model settings to values appropriate for discrete-event simulation (DES) modeling. Changes in default model settings apply to new models that you create later in the MATLAB software session.

`simeventsstartup('hybrid')` changes the default model settings to values appropriate for modeling systems that combine time-driven and discrete-event-driven behavior.

Tip To install these model settings each time you start a MATLAB software session, modify your `startup.m` file so it invokes `simeventsstartup`.

Algorithm

This function assigns the following simulation settings.

Parameter	Setting for DES Systems	Setting for Hybrid Systems
SolverName	VariableStepDiscrete	ode45
SolverType	Variable-step	Variable-step
MaxStep	inf	inf
SaveTime	off	off
SaveOutput	off	off
AlgebraicLoopMsg	error	error
SolverPrmCheckMsg	none	none

For information about the settings in the table, see “Model Parameters” in the Simulink documentation. For information about discrete and continuous solvers, see “Solvers” in the Simulink documentation.

See Also

`simeventsconfig`
`startup`

3

Block Reference

Generators (p. 3-2)	Generate entities, events, and signals
SimEvents Sinks (p. 3-3)	View and export data
Attributes (p. 3-4)	Manage data attached to entities
Queues (p. 3-5)	Store entities in queue
Servers (p. 3-6)	Delay entities by service time
Routing (p. 3-7)	Design entity paths
Gates (p. 3-8)	Regulate entity admission
Entity Management (p. 3-9)	Combine and split entities
Signal Management (p. 3-10)	Manipulate signals based on events
SimEvents Ports and Subsystems (p. 3-11)	Control timing using discrete event subsystems
Timing (p. 3-12)	Compute and limit time that entities spend in region
Event Translation (p. 3-13)	Convert event types
Probes (p. 3-14)	Report information about the simulation
SimEvents User-Defined Functions (p. 3-15)	Support custom functions

Generators

Entity Generators (p. 3-2)	Create entities
Event Generators (p. 3-2)	Create function-call events
Signal Generators (p. 3-2)	Create numerical signals

Entity Generators

Event-Based Entity Generator	Generate entity upon signal-based event or function call
Time-Based Entity Generator	Generate entities using intergeneration times from signal or statistical distribution

Event Generators

Entity-Based Function-Call Event Generator	Generate function call events corresponding to entities
Signal-Based Function-Call Event Generator	Generate function-call events in response to signal-based events

Signal Generators

Event-Based Random Number	Generate random numbers from specified distribution, parameters, and initial seed
Event-Based Sequence	Generate sequence of numbers from specified column vector

SimEvents Sinks

Attribute Scope	Plot data from attribute of arriving entities
Discrete Event Signal to Workspace	Write event-based signal to workspace
Entity Sink	Accept or block entities
Instantaneous Entity Counting Scope	Plot entity count versus time
Instantaneous Event Counting Scope	Plot event count versus time
Signal Scope	Plot data from signal
X-Y Attribute Scope	Plot data from two attributes of arriving entities
X-Y Signal Scope	Plot data from two signals

Attributes

Get Attribute	Output value of entity's attribute
Set Attribute	Assign data to entity

Queues

FIFO Queue	Store entities in sequence for undetermined length of time
LIFO Queue	Store entities in stack for undetermined length of time
Priority Queue	Store entities in sorted sequence for undetermined length of time

Servers

Infinite Server	Delay any number of entities for period of time
N-Server	Serve up to N entities for period of time
Single Server	Serve one entity for period of time

Routing

Input Switch	Accept entities from selected entity input port
Output Switch	Select entity output port for departure
Path Combiner	Merge entity paths
Replicate	Output copies of entity

Gates

Enabled Gate	Permit entity arrivals only when control signal is positive
Release Gate	Permit one pending entity to arrive when event occurs

Entity Management

Entity Combiner	Generate one entity per set of entities arriving simultaneously
Entity Splitter	Divide composite entity into component entities

Signal Management

Initial Value	Output specified value until first sample time hit
Signal Latch	Write input signal value to memory and read memory to output signal upon events

SimEvents Ports and Subsystems

Conn	Provide entity input port or entity output port for virtual subsystem
Discrete Event Inport	Input port for Discrete Event Subsystem block
Discrete Event Outport	Provide output port for Discrete Event Subsystem block
Discrete Event Subsystem	Subsystem to be executed upon signal-based events
Subsystem Configuration	Configuration for Discrete Event Subsystem block

Timing

Cancel Timeout	Cancel timeout event for each entity
Read Timer	Report statistical data about named timer associated with arriving entities
Schedule Timeout	Schedule timeout event for each entity
Start Timer	Associate named timer to each arriving entity independently and start timing

Event Translation

Entity Departure Event to Function-Call Event	Convert entity departure event into one or two function calls
Signal-Based Event to Function-Call Event	Convert signal-based events into function calls

Probes

Entity Departure Counter	Count departures and write result to signal port and/or attribute

SimEvents User-Defined Functions

Attribute Function	Access and modify attributes using Embedded MATLAB™ code

4

Blocks — Alphabetical List

Attribute Function

Purpose

Access and modify attributes using Embedded MATLAB code

Library

SimEvents User-Defined Functions

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 Embedded MATLAB subset of the MATLAB language, subject to the argument-naming rules described in "Writing Functions to Manipulate Attributes" and the attribute support described in "Attribute Value Support" in the SimEvents user guide documentation.

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

The Attribute Function block automatically ensures that the computation and the reassignment of the attribute value use the correct timing. It is not necessary to use a discrete event subsystem in conjunction with the Attribute Function block.

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

- "Example: Setting Attributes" in the SimEvents user guide documentation
- "Example: Incorporating Legacy Code" in the SimEvents user guide documentation
- "Attribute Names Unique and Accessible in Composite Entity" in the SimEvents user guide documentation
- Entity Combiner for Assembling Components demo

- Distributed Processing for Multi-Class Jobs demo, within the Distribution Center subsystem and its Service History Monitor subsystem

See Also

Set Attribute, Get Attribute

"Writing Functions to Manipulate Attributes" in the SimEvents user guide documentation

Purpose

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.

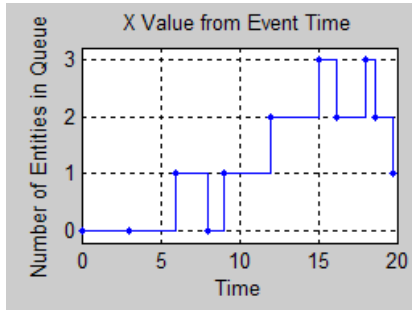
The **Plot type** parameter on the **Plotting** tab determines whether and how the block connects the points that it plots. For details, see "Connections Among Points in Plots" in the SimEvents user guide documentation.

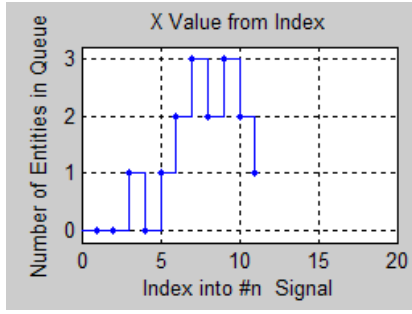
Selecting Data for the Horizontal Axis

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. This port appears only if you select **Enable entity OUT port**.

Signal Output Ports

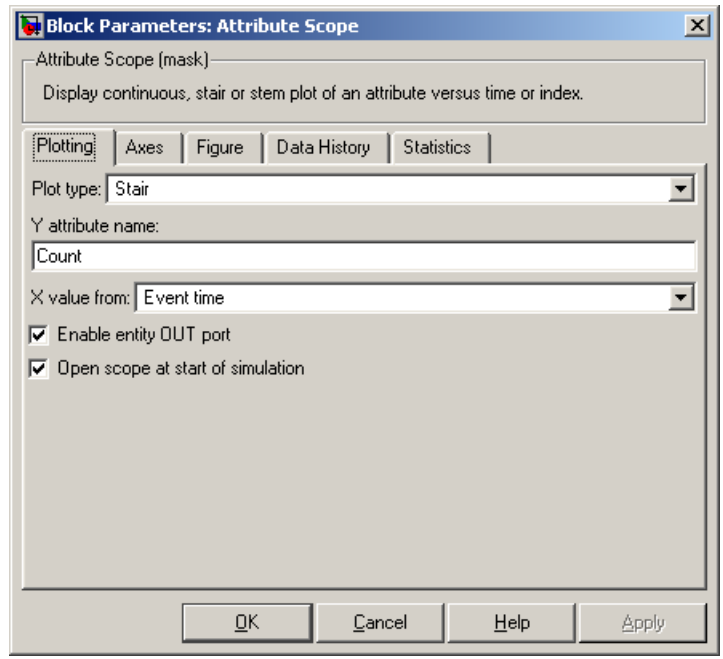
Label	Description
#a	Number of entities that have arrived at the block since the start of the simulation.

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

Dialog Box

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

Plotting Tab



Plot type

The presentation format for the data. See “Connections Among Points in Plots” in the SimEvents user guide documentation for details.

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 4-5 for details.

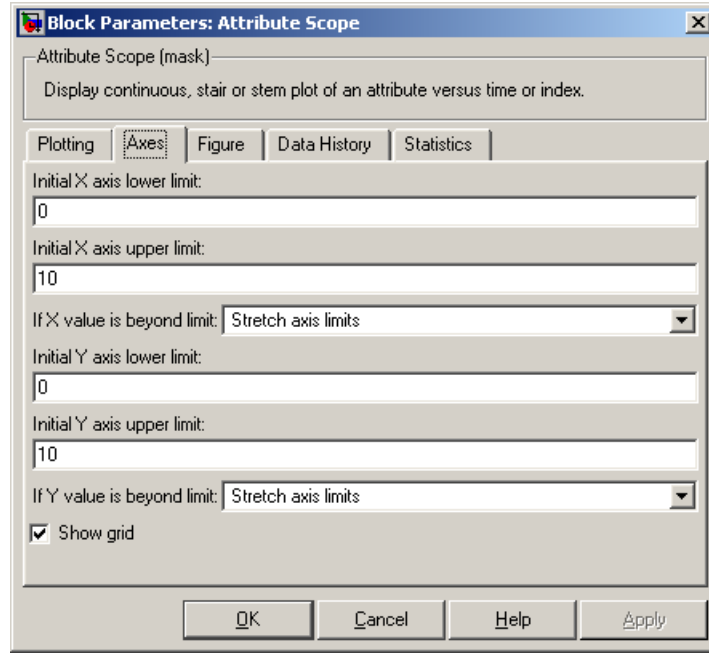
Enable entity OUT port

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.

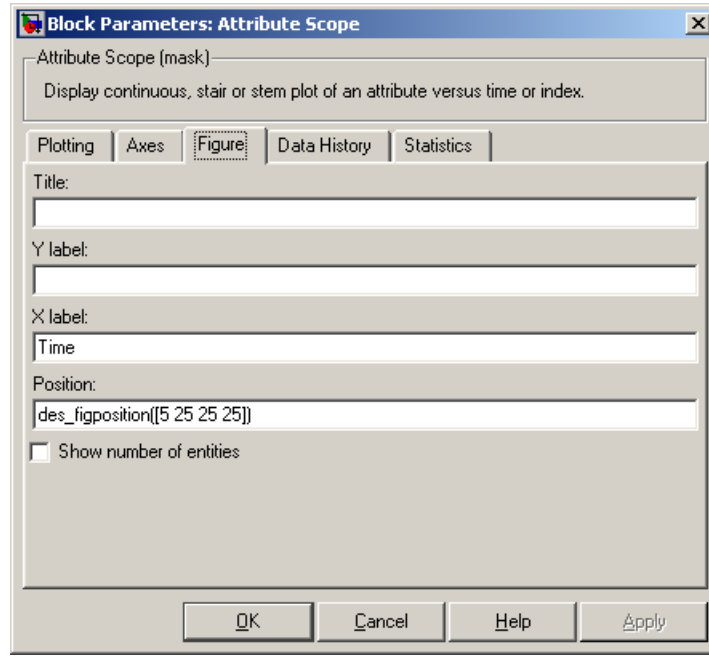
If X value is beyond limit
Determines how the plot changes if one or more X values are not within the limits shown on the X axis. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

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. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

Show grid
Toggles the grid on and off.

Figure Tab



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

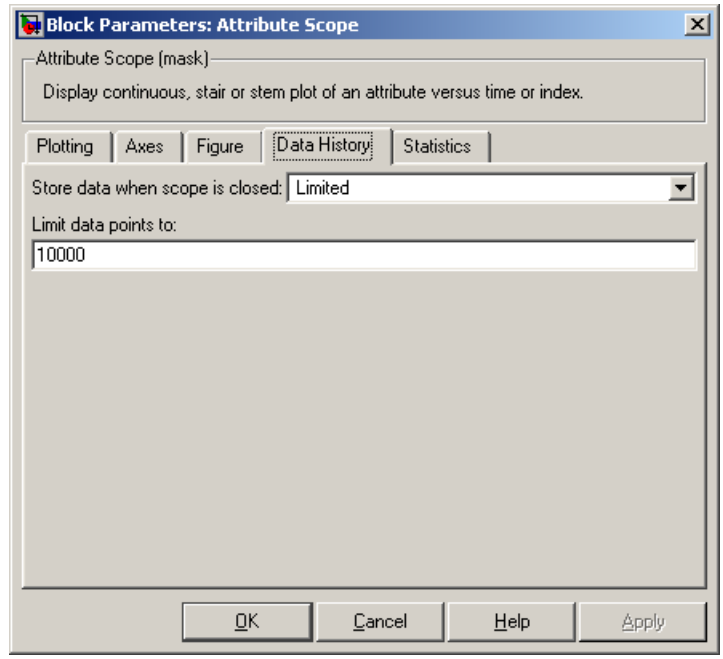
Y label
Text that appears to the left of the vertical axis.

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

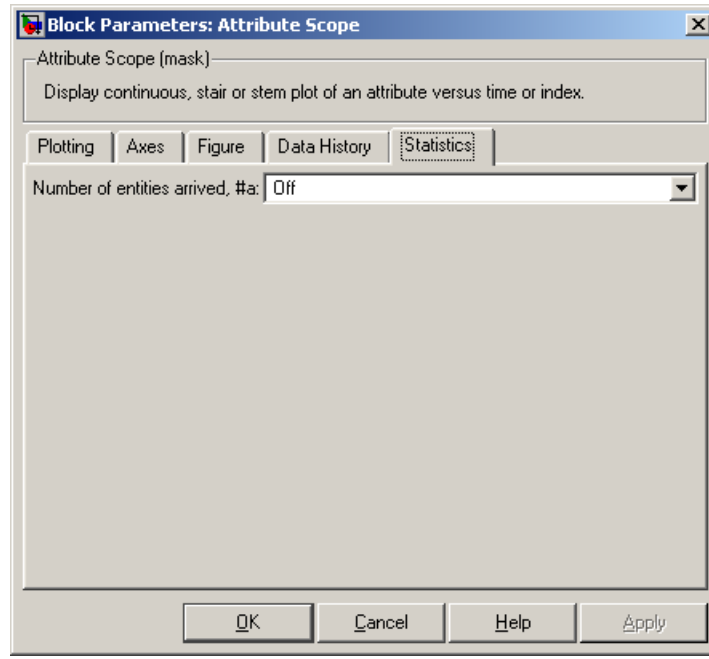


Store data when scope is closed
Select `Unlimited` to have the block cache all data for future viewing, `Limited` to cache a portion of the most recent data, and `Disabled` to avoid caching undisplayed data.

Limit data points to
The number of data points the block caches, using the most recent data. This field appears only if you set **Store data when scope is closed** to `Limited`.

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 arrived

Controls the presence and behavior of the signal output port labeled **#a**.

Examples

- “Example: Round-Robin Approach to Choosing Inputs” in the SimEvents getting started documentation
- “Example: Setting Attributes” in the SimEvents user guide documentation

See Also

X-Y Attribute Scope, Signal Scope

"Plotting Data" in the SimEvents user guide documentation, "Accessing Attributes of Entities" in the SimEvents user guide documentation

Purpose

Cancel timeout event for each entity

Library

Timing

Description

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

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

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

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

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Entity Output Ports

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

Signal Output Ports

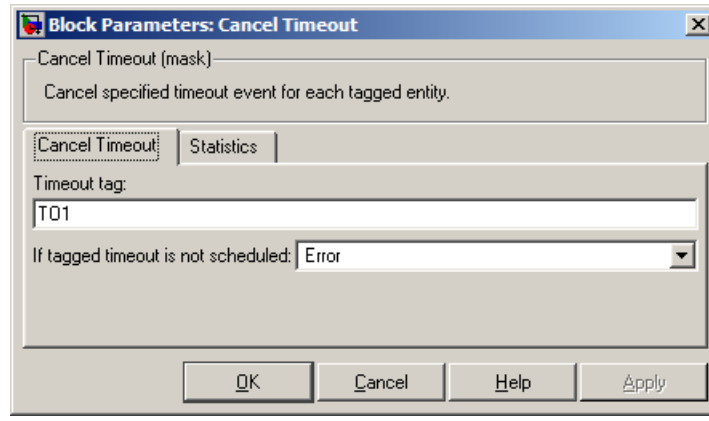
Label	Description	Time of Update When Statistic is On	Order of Update
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	3
#t	Number of entities that have departed from this block and been associated with a timeout of the specified name.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

Dialog Box

Cancel Timeout Tab



Timeout tag

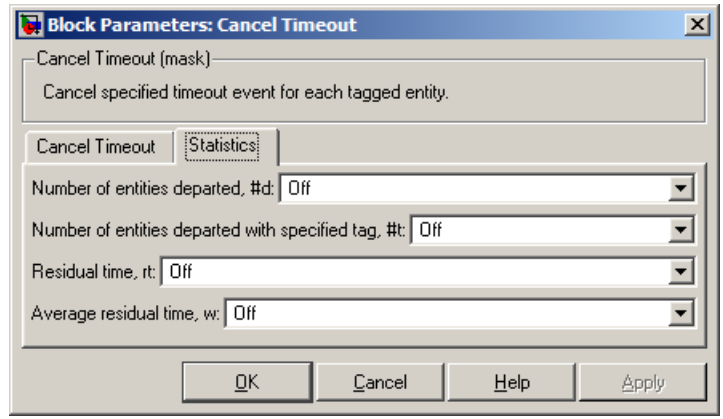
Name of the timeout event to cancel, corresponding to the **Timeout tag** parameter of a Schedule Timeout block in the model.

If tagged timeout is not scheduled

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

Statistics Tab

These parameters determine whether 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
Controls the presence and behavior of the signal output port labeled **#d**.

Number of entities departed with specified tag
Controls the presence and behavior of the signal output port labeled **#t**.

Residual time
Controls the presence of the signal output port labeled **rt**.

Average residual time
Controls the presence and behavior of the signal output port labeled **w**.

Examples

- "Basic Example Using Timeouts" in the SimEvents user guide documentation
- "Defining Entity Paths on Which Timeouts Apply" in the SimEvents user guide documentation
- "Example: Dropped and Timed-Out Packets" in the SimEvents user guide documentation
- "Example: Rerouting Timed-Out Entities to Expedite Handling" in the SimEvents user guide documentation

- “Example: Limiting the Time Until Service Completion” in the SimEvents user guide documentation

See Also

Schedule Timeout

“Forcing Departures Using Timeouts” in the SimEvents user guide documentation

Conn

Purpose

Provide entity input port or entity output port for virtual subsystem

Library

SimEvents Ports and Subsystems

Description

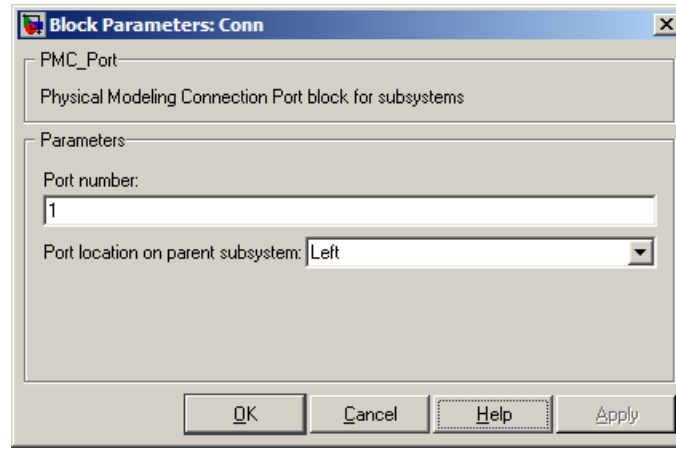
The Conn block, placed inside a subsystem containing blocks with entity ports, creates an entity port on the boundary of the subsystem. When you connect the Conn block, the port changes its appearance and becomes either an entity input port or an entity output port:

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

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

Note This block is for entity paths in virtual subsystems, not signal connections in discrete event subsystems. To add a port to the Discrete Event Subsystem block, use the Discrete Event Inport or Discrete Event Outport block.

Dialog Box



Port number

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

Port location on parent subsystem

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

Discrete Event Inport

Purpose

Input port for Discrete Event Subsystem block

Library

SimEvents Ports and Subsystems

Description

Discrete Event Inport blocks are the links from outside a discrete event subsystem into the subsystem. The application executes the subsystem when at least one of the Discrete Event Inport blocks within the subsystem detects a qualifying signal-based event. If the signal corresponding to this block is a nonscalar array, the block detects a single qualifying event if any of the positions in the array has a qualifying event; for example, a value change from `[1 2 3]` to `[1 5 6]` represents one event, not two events. If N distinct qualifying events occur at the same time at distinct Discrete Event Inport blocks or at distinct sample time hits of a single Discrete Event Inport block, then the subsystem executes N times and updates its output signals N times.

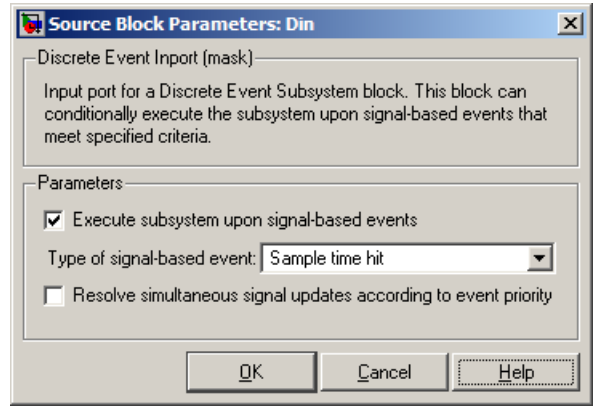
By default, the names of Discrete Event Inport blocks appear in the subsystem window as **Din**, **Din1**, **Din2**, and so on. A discrete event inport represents a real or complex sample-based signal of type `double` and any dimension.

Copying and pasting Discrete Event Inport blocks is supported, but duplicating them is not.

To create a subsystem using the Discrete Event Subsystem block, see "Setting Up Signal-Based Discrete Event Subsystems" in the SimEvents user guide documentation.

Note This block is for signal connections in discrete event subsystems, not entity paths in virtual subsystems. To add a port to a virtual subsystem along an entity path, use the Conn block.

Dialog Box



Execute subsystem upon signal-based events

If you select this option, the application executes the subsystem when a qualifying signal-based event occurs in the signal corresponding to this inport block. If you clear this option, the subsystem reads the signal upon execution but does not respond to its events.

Type of signal-based event

Determines the type of event that is a qualifying event in the signal corresponding to this inport block. If the signal is complex, you must select `Sample time hit`. This field appears only if you select **Execute subsystem upon signal-based events**.

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

The type of change in the control signal's value, or the type of trigger, that further restricts the event type specified in **Type of signal-based event**. This field appears only if you set **Type of signal-based event** to `Change in signal` or `Trigger`.

Resolve simultaneous signal updates according to event priority

Select this option to control the sequencing of the subsystem's execution event in response to updates in the signal corresponding to this inport 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 that causes it. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation.

Event priority
The priority of the subsystem's execution event (in response to updates in the signal corresponding to this inport block), relative to other simultaneous events in the simulation. For details, see "Specifying Event Priorities to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears only if you select **Resolve simultaneous signal updates according to event priority**.

Examples

See "Examples Using Discrete Event Subsystem Blocks" in the SimEvents user guide documentation.

See Also

Discrete Event Subsystem, Discrete Event Outport

"Controlling Timing with Subsystems" in the SimEvents user guide documentation

Purpose

Provide output port for Discrete Event Subsystem block

Library

SimEvents Ports and Subsystems

Description

Discrete Event Outport blocks are the links from a discrete event subsystem to a destination outside the subsystem. By default, the names of Discrete Event Outport blocks appear in the subsystem window as **Dout**, **Dout1**, **Dout2**, and so on.

To create a subsystem using the Discrete Event Subsystem block, see "Setting Up Signal-Based Discrete Event Subsystems" in the SimEvents user guide documentation.

Note This block is for signal connections in discrete event subsystems, not entity paths in virtual subsystems. To add a port to a virtual subsystem along an entity path, use the Conn block.

Examples

See "Examples Using Discrete Event Subsystem Blocks" in the SimEvents user guide documentation.

See Also

Discrete Event Subsystem, Discrete Event Inport

"Controlling Timing with Subsystems" in the SimEvents user guide documentation

Discrete Event Signal to Workspace

Purpose

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. 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.

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 in the Simulink documentation.

Comparison with To Workspace Block

This block can detect zero-duration values of the input signal, as well as signal updates that do not necessarily correspond to time steps determined by time-based dynamics.

This block does not support data types other than `double`, and 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

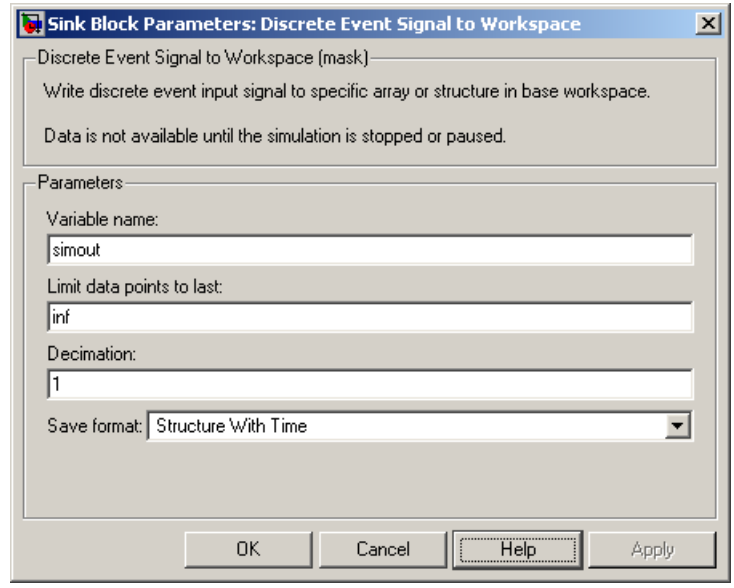
Configuration Parameters dialog box. By default, this option is selected and the variable is called `tout`. The `simeventsconfig` function clears the time logging option to avoid confusion between the time steps listed in `tout` and the update times of event-based signals in the simulation.

Ports

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

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

Dialog Box



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

Limit data points to last
The maximum number of input samples to be saved.

Decimation
A positive integer, n, that specifies the decimation factor. The block ignores the first 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

- "Example: Sending Queue Length to the Workspace" in the SimEvents user guide documentation
- "Example: Observing Service Completions" in the SimEvents user guide documentation

See Also

To Workspace

"Sending Data to the MATLAB Workspace" in the SimEvents user guide documentation

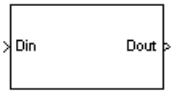
Purpose

Subsystem to be executed upon signal-based events

Library

SimEvents Ports and Subsystems

Description



This block represents a subsystem of the system that contains it. It is configured so that the application executes the subsystem when at least one of the Discrete Event Inport blocks within the subsystem detects a qualifying signal-based event. If N qualifying events occur at the same simulation time (whether at distinct Discrete Event Inport blocks or at distinct sample time hits of a single Discrete Event Inport block), then the subsystem executes N times and updates its output signals N times.

Attach inputs from the upper level to blocks inside the subsystem using the **Din**, **Din1**, **Din2**, and similarly labeled ports on the Discrete Event Subsystem block. Inputs are real or complex sample-based signals, of type `double` and any dimension. Attach outputs, if necessary, from blocks inside the subsystem to the upper level using the **Dout**, **Dout1**, **Dout2**, and similarly labeled ports on the Discrete Event Subsystem block.

The number of input ports drawn on the Discrete Event Subsystem block's icon corresponds to the number of nonduplicate Discrete Event Inport blocks inside the subsystem. Similarly, the number of output ports drawn on the block corresponds to the number of Discrete Event Outport blocks inside the subsystem.

Note This block is compatible only with inports and outports from the SimEvents Ports and Subsystems library. To add inports or outports to the Discrete Event Subsystem window, either copy new ports from the SimEvents Ports and Subsystems library (not the Simulink Ports & Subsystems library), or copy and paste the ports that are in the Discrete Event Subsystem window by default.

To create a subsystem using the Discrete Event Subsystem block, see "Setting Up Signal-Based Discrete Event Subsystems" in the SimEvents user guide documentation. A discussion of discrete event

subsystems and examples using this block are in "Controlling Timing with Subsystems" in the SimEvents user guide documentation. To view the contents of the subsystem, double-click the Discrete Event Subsystem block; the Model Explorer tool does not show the contents.

"Block execution" in this documentation is shorthand for "block methods execution." Methods are functions that the Simulink engine uses to solve a model. Blocks are made up of multiple methods. For details, see "Block Methods" in the Simulink documentation.

Ports

Signal Input Ports

Label	Description
Din, **Din1**, **Din2**, and so on	Signals that serve as inputs to the blocks in the subsystem. At least one input signal must be present; others are optional. Each port appears only if the subsystem contains a Discrete Event Inport block of the same name.

Signal Output Ports

Label	Description
Dout, **Dout1**, **Dout2**, and so on	Optional signals that serve as outputs from the blocks in the subsystem. Each port appears only if the subsystem contains a Discrete Event Outport block of the same name.

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

Examples

See "Examples Using Discrete Event Subsystem Blocks" in the SimEvents user guide documentation.

See Also

Discrete Event Inport, Discrete Event Outport

"Controlling Timing with Subsystems" in the SimEvents user guide documentation

Enabled Gate

Purpose

Permit entity arrivals only when control signal is positive

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.

Entity Output Ports

Label	Description
OUT	Port for departing entities.

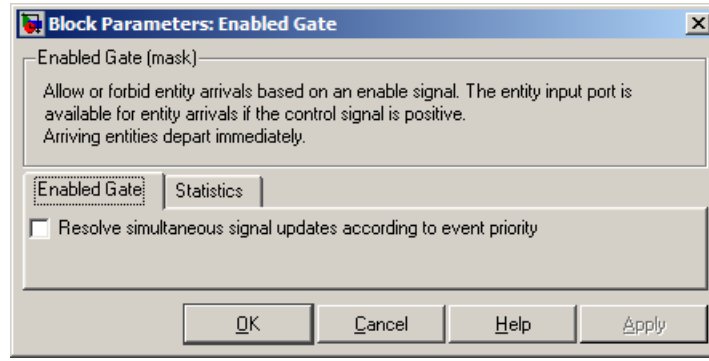
Signal Output Ports

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, which is in effect from the start of the simulation until the first update by the block, is 0.

Dialog Box

Enabled Gate Tab



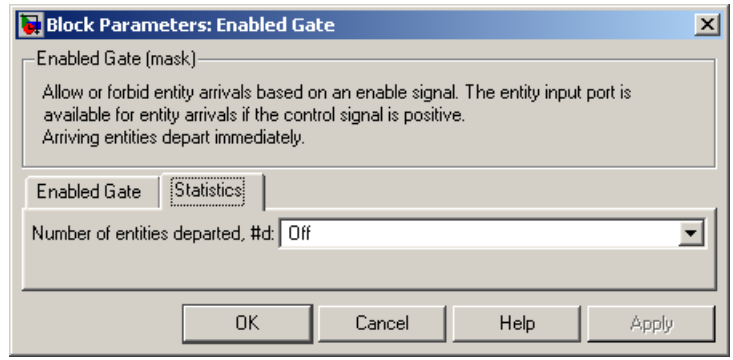
Resolve simultaneous signal updates according to event priority
Select this option to explicitly prioritize the gate-opening or gate-closing event, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority `SYS1` on the event calendar. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation.

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. For

details, see “Specifying Event Priorities to Resolve Simultaneous Signal Updates” in the SimEvents user guide documentation. This field appears only if you select **Resolve simultaneous signal updates according to event priority**.

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

Controls the presence and behavior of the signal output port labeled **#d**.

Examples

- “Example: Controlling Joint Availability of Two Servers” in the SimEvents user guide documentation
- “Example: First Entity as a Special Case” in the SimEvents user guide documentation
- “Example: Limiting the Time Until Service Completion” in the SimEvents user guide documentation

See Also

Release Gate

"Regulating Arrivals Using Gates" in the SimEvents user guide documentation

Entity Combiner

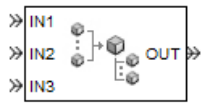
Purpose

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.

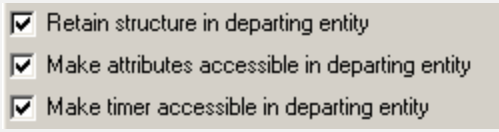
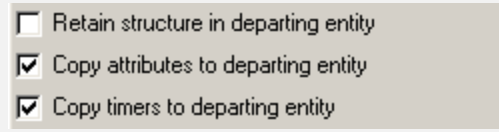
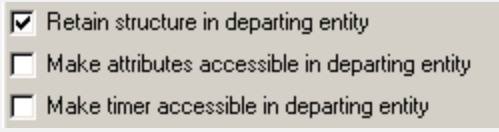
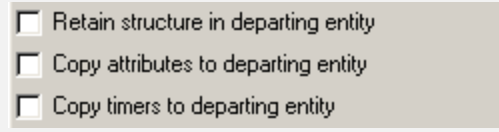
Managing Information When Combining Entities

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

 • 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.	 • You cannot split the departing entity. • Attribute names and timer tags must be unique. • You can access attributes and timers.
 • 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.	 • 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.

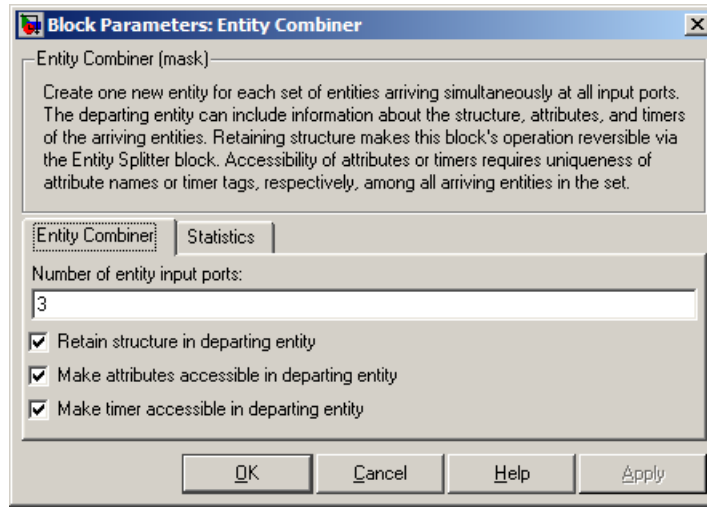
Signal Output Ports

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, which is in effect from the start of the simulation until the first update by the block, is 0.

Dialog Box

Entity Combiner Tab



Number of entity input ports

Determines how many entity input ports the block has. This parameter must be a literal value, not a variable or an expression requiring evaluation.

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.

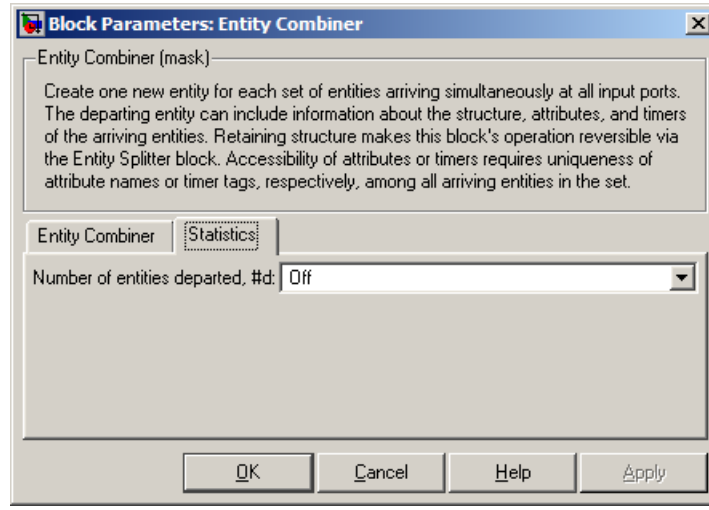
Make attributes accessible in departing entity, **Copy attributes to departing entity**

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, **Copy timers to 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

Controls the presence and behavior of the signal output port labeled **#d**.

Examples

- "Example: Waiting to Combine Entities" in the SimEvents user guide documentation
- "Example: Copying Timers When Combining Entities" in the SimEvents user guide documentation
- "Example: Managing Data in Composite Entities" in the SimEvents user guide documentation
- Entity Combiner for Assembling Components demo

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

"Combining Entities" in the SimEvents user guide documentation

Purpose

Count departures and write result to signal port and/or attribute

Library

Probes

Description

This block computes the number of entities that have departed from this block 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 departing entity is included in the count.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Signal Input Ports

Label	Description
tr	When the signal at this port satisfies the specified trigger criteria, the block resets its internal counter and the **#d** output signal to zero. This port appears only if you set **Reset counter upon** to `Trigger from port tr`.
vc	When the signal at this port satisfies the specified value-change criteria, the block resets its internal counter and the **#d** output signal to zero. This port appears only if you set **Reset counter upon** to `Change in signal from port vc`.

Entity Output Ports

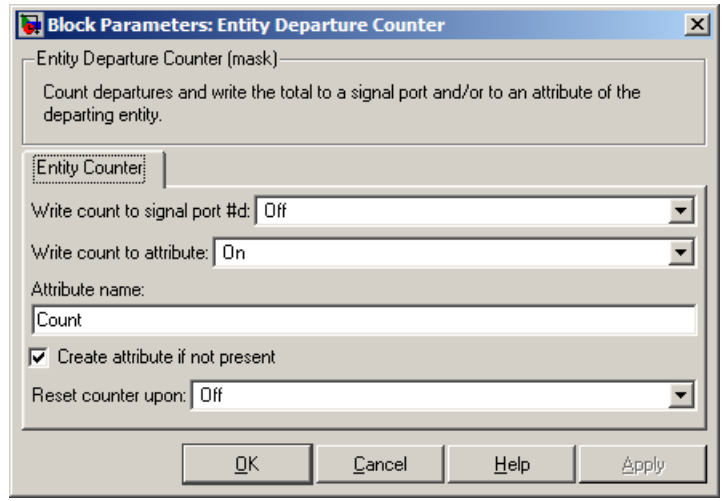
Label	Description
OUT	Port for departing entities.

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, which is in effect from the start of the simulation until the first update by the block, is 0.

Dialog Box



Write count to signal port #d

Controls the presence and behavior of the signal output port labeled **#d**. This parameter determines whether the block outputs the entity count through a signal output port throughout the simulation, only when you stop or pause the simulation, or not at all.

Write count to attribute
If you set this to `On`, 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. This field appears only if you set **Write count to attribute** to `On`.

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 named above does not already exist. This field appears only if you set **Write count to attribute** to `On`.

Reset counter upon
Determines whether the block resets its internal counter to zero based on trigger edges, based on changes in signal values, or not at all.

Trigger type
Determines whether rising, falling, or either type of trigger edge causes the counter to reset. This field appears 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. This field appears 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 explicitly prioritize the counter's reset event, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority `SYS1` on the event calendar. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears only if you set **Reset counter upon** to `Trigger from port tr` or `Change in signal from port vc`.

Event priority

The priority of the counter's reset event, relative to other simultaneous events in the simulation. For details, see "Specifying Event Priorities to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears only if you set **Reset counter upon** to either `Trigger from port tr` or `Change in signal from port vc`, and then select **Resolve simultaneous signal updates according to event priority**.

Examples

- "Example: Setting Attributes" in the SimEvents user guide documentation
- "Example: Resetting a Counter After a Transient Period" in the SimEvents user guide documentation
- "Stopping Based on Entity Count" in the SimEvents user guide documentation

See Also

Instantaneous Entity Counting Scope

"Counting Entities" in the SimEvents user guide documentation

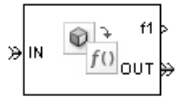
Purpose

Convert entity departure event into one or two function calls

Library

Event Translation

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 input port appears 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 input port appears 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

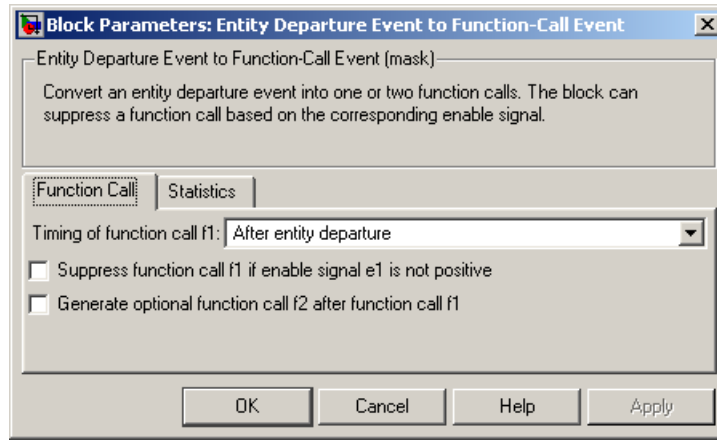
Label	Description	Time of Update When Port Is Present	Order of Update
f1	Function-call signal, possibly contingent on **e1** input signal.	Before or after entity departure, depending on **Timing of function call f1** parameter	1
f2	Function-call signal, possibly contingent on **e2** input signal.	Before 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.

Dialog Box

Function Call Tab



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

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.

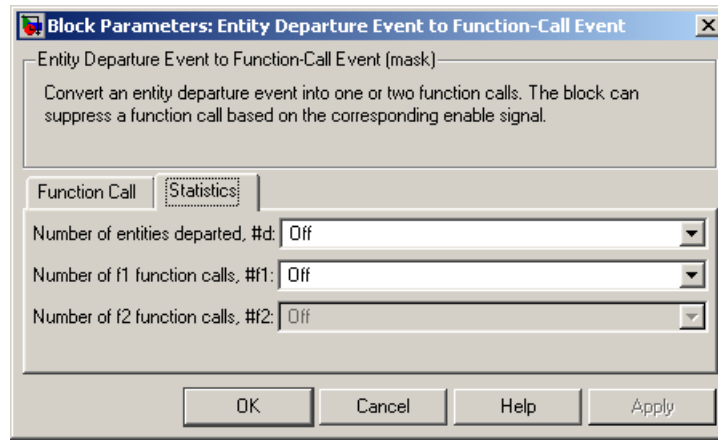
Generate optional function call f2 after function call f1
Selecting this option causes the block to generate a function call at the optional **f2** output port when appropriate criteria are satisfied.

Timing of function call f2
Determines whether the **f2** function call occurs before or after the entity departure event. This field appears 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. This field appears

only if you select **Generate optional function call f2 after function call f1**.

Statistics Tab



Number of entities departed
Controls the presence and behavior of the signal output port labeled **#d**.

Number of f1 function calls
Controls the presence and behavior of the signal output port labeled **#f1**.

Number of f2 function calls
Controls the presence and behavior of 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.

Examples

- "Example: Opening a Gate Upon Entity Departures" in the SimEvents user guide documentation
- "Example: Using Entity-Based Timing for Choosing a Port" in the SimEvents user guide documentation

- Markov-Modulated Poisson Process demo, within On-Off Modulated Markov Source subsystems

See Also

Signal-Based Event to Function-Call Event

"Manipulating Events" in the SimEvents user guide documentation

Purpose

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.

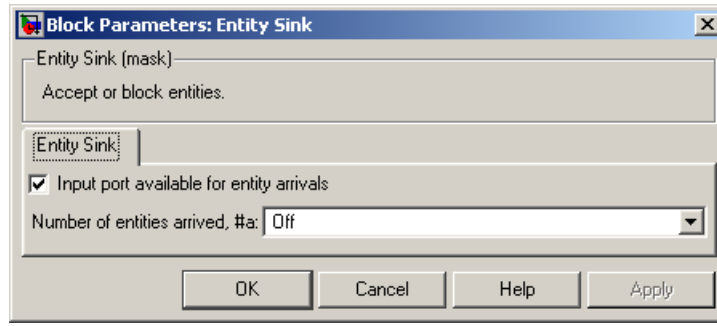
Signal Output Ports

Label	Description	Time of Update When Statistic Is On
#a	Number of entities that the block has accepted. This port appears only if you select **Input port available for entity arrivals**, and then set **Number of entities arrived** to either `On` or `Upon stop or pause`.	After entity arrival

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

Entity Sink

Dialog Box



Input port available for entity arrivals

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

Number of entities arrived

Controls the presence and behavior of the signal output port labeled **#a**. This field appears only if you select **Input port available for entity arrivals**.

Examples

- "Modeling the Channels" in the SimEvents getting started documentation
- "Example: Using an Attribute to Select an Output Port" in the SimEvents getting started documentation

See Also

Time-Based Entity Generator, Event-Based Entity Generator

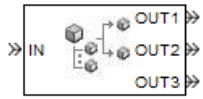
Purpose

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 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 block instead of the Entity Splitter block. The Replicate block copies entities without regard to their structure.

Attributes and Timers

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

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

Complete or Partial Splitting

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

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

Departure of Component Entities

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

Parameter Value	Description	Example
`OUT1 port`	Each time the block splits an entity, the component entities depart via entity output ports **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.
`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.

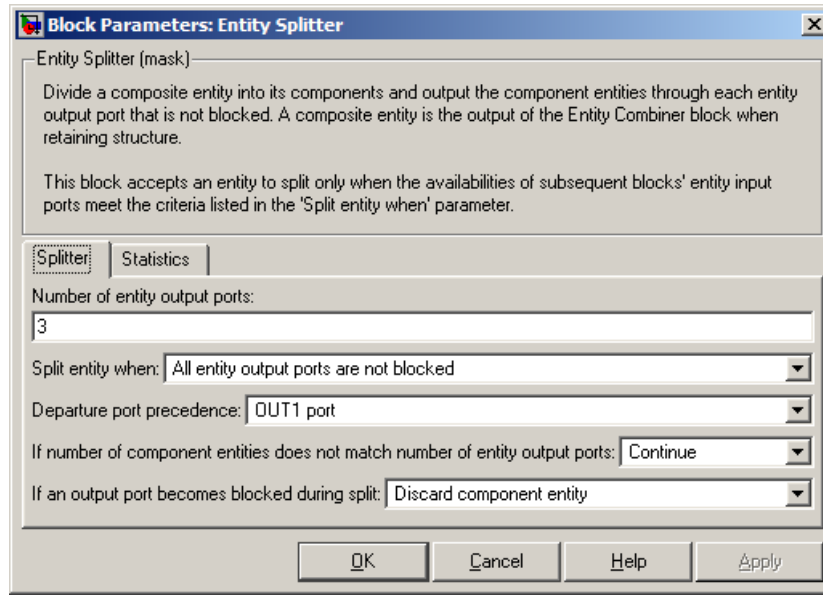
Signal Output Ports

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.

Dialog Box

Entity Splitter Tab



Number of entity output ports

Determines how many entity output ports the block has. This parameter must be a literal value, not a variable or an expression requiring evaluation.

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. This field

appears only if you set **Departure port precedence** to `Equiprobable`.

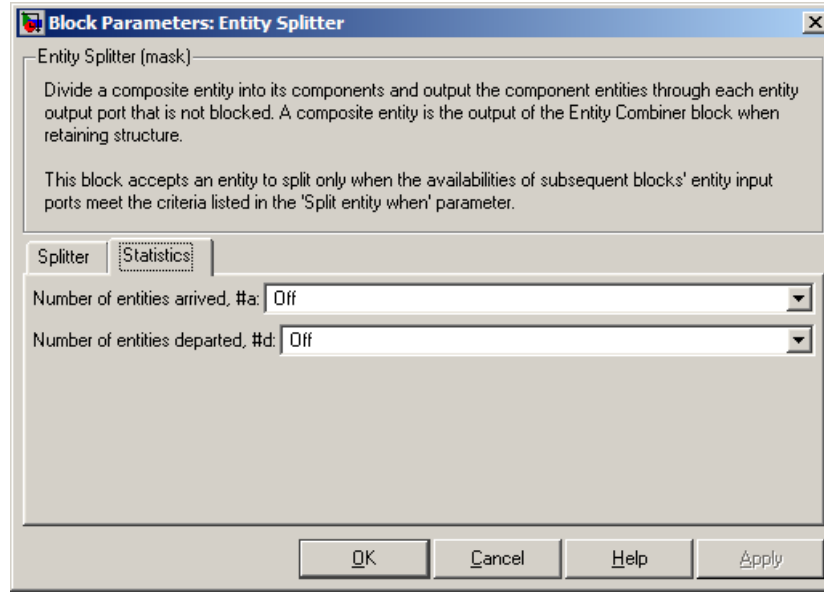
If number of component entities does not match number of entity output ports

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. This field appears only if you set **Split entity when** to `All entity output ports are not blocked`.

Statistics Tab



Number of entities arrived
Controls the presence and behavior of the signal output port labeled **#a**.

Number of entities departed
Controls the presence and behavior of the signal output port labeled **#d**.

Examples

See "Example: Managing Data in Composite Entities" in the SimEvents user guide documentation.

See Also

Entity Combiner

"Combining Entities" in the SimEvents user guide documentation

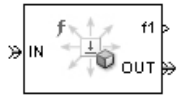
Purpose

Generate function call events corresponding to entities

Library

Generators / Event Generators

Description



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 Event to Function-Call Event 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

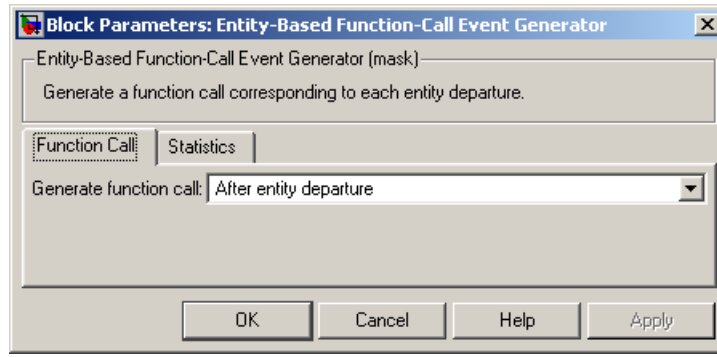
Signal Output Ports (Continued)

Label	Description	Time of Update When Port Is Present	Order of Update
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	3
#f1	Number of function calls the block has generated since the start of the simulation.	After entity departure	2

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

Dialog Box

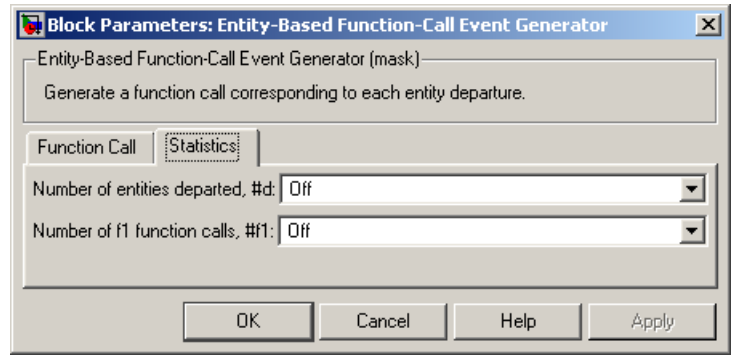
Function Call Tab



Generate function call

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

Statistics Tab



Number of entities departed

Controls the presence and behavior of the signal output port labeled **#d**.

Number of f1 function calls

Controls the presence and behavior of the signal output port labeled **#f1**.

Examples

See "Example: Performing a Computation on Selected Entity Paths" in the SimEvents user guide documentation.

See Also

Entity Departure Event to Function-Call Event, Signal-Based Function-Call Event Generator

"Generating Function-Call Events" in the SimEvents user guide documentation

Event-Based Entity Generator

Purpose

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 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 generates 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 4-66.

Responding to Blockage at the Entity Output Port

Using the **Allow OUT port blocking** and optional **Response during blockage period** parameters, you can choose how this block responds when the subsequent entity input port is not available to accept the newly generated entity. The possible responses are in the table below.

Response to Blockage	Parameter Values
Error message	Clear the **Allow OUT port blocking** check box.

Response to Blockage	Parameter Values
The block discards all entities generated while the **OUT** port is blocked. Such entities do not depart from the block.	Select **Allow OUT port blocking** and set **Response during blockage period** to `Discard generated entities`
The block holds one generated entity and temporarily suspends the generation of additional entities. When the subsequent entity input port becomes available, this block outputs the held entity and 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	Signal whose updates indicate when to generate entities. This port appears only if you set **Generate entities upon** to `Sample time hit from port ts`.
tr	Trigger signal whose edges indicate when to generate entities. This port appears only if you set **Generate entities upon** to `Trigger from port tr`.
vc	Signal whose numerical changes in value indicate when to generate entities. This port appears only if you set **Generate entities upon** to `Change in signal from port vc`.
fcn	Function-call signal that indicates when to generate entities. This port appears only if you set **Generate entities upon** to `Function call from port fcn`.

Entity Output Ports

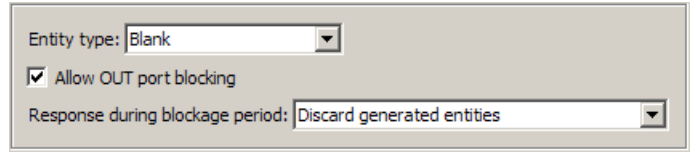
Label	Description
OUT	Port through which generated entities depart.

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 when the block tries and fails to output an entity.	After entity generation if **OUT** port is blocked, and when entity departure is imminent in all cases	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.

Dialog Box

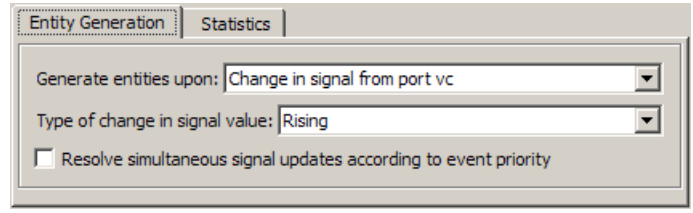


Entity type
The blank type includes no attributes. The standard type includes attributes called `Priority` and `Count` with default values of `10` and `0`, respectively.

Allow OUT port blocking
If you do not select this option, the simulation halts with an error message if the subsequent entity input port is not available to accept an entity when this block generates it.

Response during blockage period
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 4-66. This field appears only if you select **Allow OUT port blocking**.

Entity Generation Tab



Generate entities upon
The type of event that indicates when the block should generate an entity.

Trigger type
Determines whether rising, falling, or either type of trigger edge causes an entity generation. This field appears 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. This field appears only if you set **Generate entities upon** to `Change in signal from port vc`.

Resolve simultaneous signal updates according to event priority
Select this option to explicitly prioritize the entity-generation event, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority `SYS1` on the event calendar. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation.

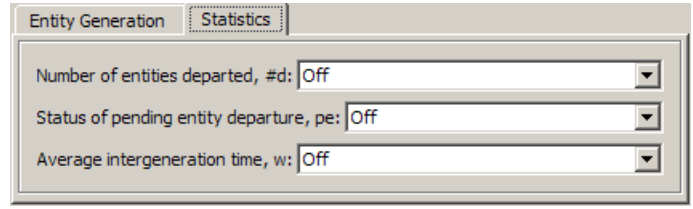
Event priority
The priority of the entity-generation event, relative to other simultaneous events in the simulation. For details, see "Specifying Event Priorities to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears only if you select **Resolve simultaneous signal updates according to event priority**.

Generate entity at simulation start
If you select this option, 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. This field appears only if you set **Generate entities** to `Sample time hit from port ts`.

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

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
: Controls the presence and behavior of the signal output port labeled **#d**.

Status of pending entity departure
: Controls the presence and behavior of the signal output port labeled **pe**.

Average intergeneration time
: Controls the presence and behavior of the signal output port labeled **w**.

Examples

- "Example: Plotting Event Counts to Check for Simultaneity" in the SimEvents user guide documentation
- "Example: Choices of Values for Event Priorities" in the SimEvents user guide documentation
- "Example: Varying Fluid Flow Rate Based on Batching Logic" in the SimEvents user guide documentation
- "Sample Use Cases for Event-Based Generation of Entities" in the SimEvents user guide documentation

See Also

Time-Based Entity Generator, Entity Sink

"Generating Entities When Events Occur" in the SimEvents user guide documentation

Event-Based Random Number

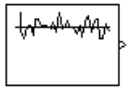
Purpose

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 is designed to infer from a subsequent block when to generate a new random number.

Connections from the Event-Based Random Number block must satisfy all of these conditions:

- Exactly one line must connect to a port listed in “Notifying Ports” in the SimEvents user guide documentation
- Zero or more lines may connect to ports listed in “Monitoring Ports” in the SimEvents user guide documentation
- No lines may connect to other ports; in particular, do not connect this block to the ports listed in “Reactive Ports” in the SimEvents user guide documentation

Note Connections from the Event-Based Random Number block to ports other than notifying ports and monitoring ports are not supported. To create a random signal that can be an input to a reactive port, see the techniques described in "Generating Random Signals Based on Arbitrary Events" and "Generating Random Time-Based Signals" in the SimEvents user guide documentation.

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**
`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**

Distribution	Additional Parameters
`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 4-82 below.

Range of Output Values

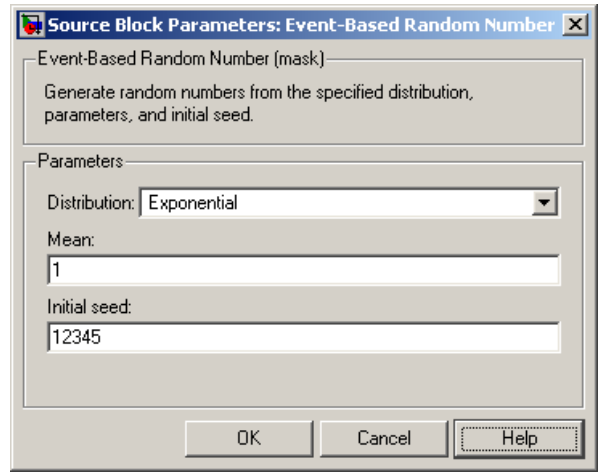
Different distributions have different output ranges. Make sure the distribution and parameters you choose are suitable for your application. For example, when generating random service times, do not use a Gaussian distribution because it can produce negative numbers.

Ports

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

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

Dialog Box



Distribution
The distribution from which the block generates random numbers.

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

Minimum, **Maximum**
The minimum and maximum values of a uniform, triangular, beta, or discrete uniform distribution.

Probability for output to be 0
The probability of a zero 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.

Examples

See "Examples of Random Event-Based Signals" in the SimEvents user guide documentation.

Algorithm

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

Exponential Distribution

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

where μ is the **Mean** parameter, a positive number.

A similar function in the Statistics Toolbox™ software is `exprnd`.

Uniform Distribution

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

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

Similar functions are `rand` in MATLAB software and `unifrnd` in the Statistics 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 Toolbox software is `binornd`.

Triangular Distribution

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

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

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

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).

A similar function in the Statistics 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 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 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 Toolbox software is `poissrnd`.

Lognormal Distribution

$$f(x) = \begin{cases} \dfrac{\exp\left[\dfrac{-\left(\ln(x-\theta)-\mu\right)^{2}}{2\sigma^{2}}\right]}{(x-\theta)\sigma\sqrt{2\pi}} & \text{for } x \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

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 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

$$f(x) = \begin{cases} \dfrac{(x-L)^{a-1}(U-x)^{b-1}}{B(a,b)(U-L)^{a+b+1}} & \text{for } L \le x \le U \\ 0 & \text{otherwise} \end{cases}$$

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 Toolbox software is `betarnd`.

Discrete Uniform Distribution

$$f(x) = \begin{cases} 1/K & \text{for } x = L + k\dfrac{(U-L)}{K-1},\ k = 0,1,2,\ldots,K-1 \\ 0 & \text{otherwise} \end{cases}$$

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 Toolbox software.

Weibull Distribution

$$f(x) = \begin{cases} \frac{\gamma}{\alpha}\left(\frac{x-\theta}{\alpha}\right)^{(\gamma-1)} \exp\left[-\left(\frac{x-\theta}{\alpha}\right)^{\gamma}\right] & \text{for } x \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where 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 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, Event-Based Sequence

"Generating Random Signals" in the SimEvents user guide documentation

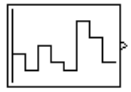
Purpose

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.

Parameter Value	Description
`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

The Event-Based Sequence block has a restricted set of valid connections to other blocks because it is designed to infer from a subsequent block when to generate a new number.

Connections from the Event-Based Sequence block must satisfy all of these conditions:

- Exactly one line must connect to a port listed in “Notifying Ports” in the SimEvents user guide documentation
- Zero or more lines may connect to ports listed in “Monitoring Ports” in the SimEvents user guide documentation
- No lines may connect to other ports; in particular, do not connect this block to the ports listed in “Reactive Ports” in the SimEvents user guide documentation

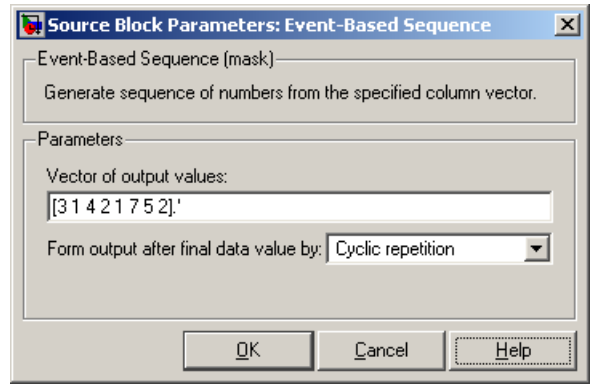
Note Connections from the Event-Based Sequence block to ports other than notifying ports and monitoring ports are not supported. To create a sequence that can be an input to a reactive port, see the techniques described in "Generating Random Signals Based on Arbitrary Events" in the SimEvents user guide documentation or use time-based blocks such as Repeating Sequence Stair and From Workspace.

Ports

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

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

Dialog Box



Vector of output values

A column vector whose entries become values of this block's output signal. To use a column vector variable in the MATLAB base workspace, enter the variable name.

Form output after final data value by

The method for generating output after the block exhausts the data referenced in the **Vector of output values** parameter.

Examples

- "Specifying Generation Times for Entities" in the SimEvents user guide documentation
- "Example: Counting Simultaneous Departures from a Server" in the SimEvents user guide documentation
- "Example: Setting Attributes" in the SimEvents user guide documentation

See Also

Event-Based Random Number, Repeating Sequence Stair, From Workspace

"Using Data Sets to Create Event-Based Signals" in the SimEvents user guide documentation

Purpose

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 and N>0. 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; see "Forcing Departures Using Timeouts" in the SimEvents user guide documentation for details about timeouts. The length of time that an entity stays in this block cannot be determined in advance.

The **IN** port is unavailable whenever N>0 and this block stores exactly N entities. In this case, the queue is said to be full.

If N=0, then the availability of the **IN** port matches the availability of the subsequent block's entity input port to which this block's **OUT** port is connected. If N=0, all arriving entities depart immediately because this block cannot store entities.

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. This port appears 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 queue.	After entity arrival and after entity departure	3
pe	A value of 1 indicates when the block tries and fails to output an entity. A value of 0 indicates when the queue is empty or an entity departure is imminent.	After entity arrival and when entity departure is imminent	1
w	Sample mean of the waiting times in this block for all entities that have departed via any port.	After entity departure	2

Signal Output Ports (Continued)

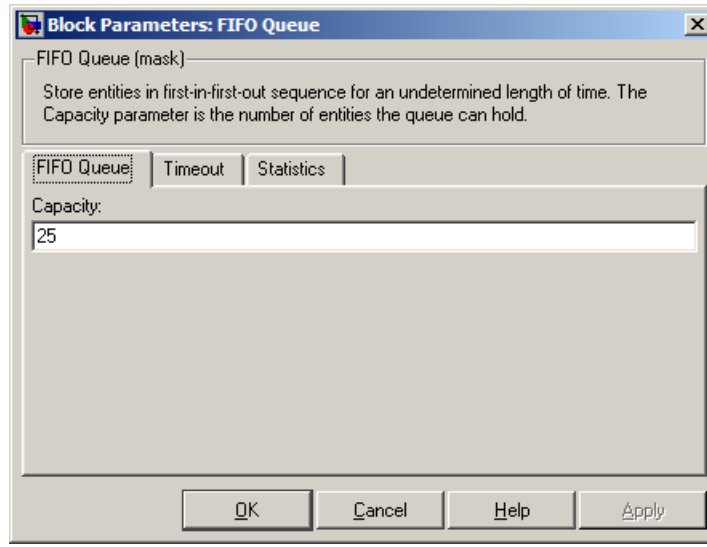
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
len	Average number of entities in the queue over time, that is, the time average of the **#n** signal.	After entity arrival and after entity departure. If the entity arrives at an empty queue and immediately departs, then **len** has one sample time hit, not two.	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.

Dialog Box

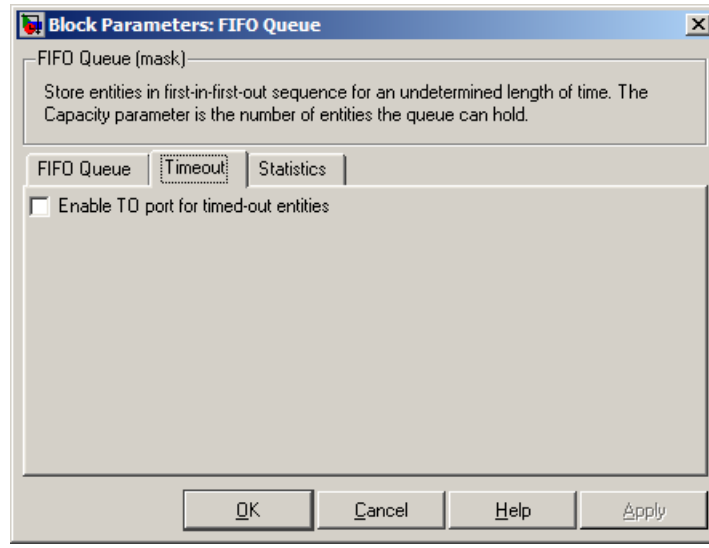
FIFO Queue Tab



Capacity

Determines how many entities the block can store at a time.

Timeout Tab

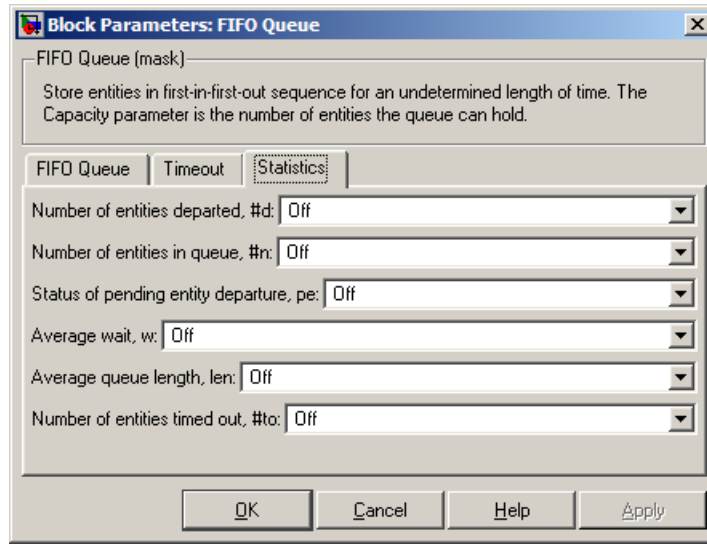


Enable TO port for timed-out entities

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 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

Controls the presence and behavior of the signal output port labeled **#d**.

Number of entities in queue

Controls the presence and behavior of the signal output port labeled **#n**.

Status of pending entity departure

Controls the presence of the signal output port labeled **pe**.

Average wait

Controls the presence and behavior of the signal output port labeled **w**.

Average queue length

Controls the presence and behavior of the signal output port labeled **len**.

Number of entities timed out
Controls the presence and behavior of the signal output port labeled **#to**.

Examples

- "Building a Simple Discrete-Event Model" in the SimEvents getting started documentation
- "Example: Selecting the First Available Server" in the SimEvents getting started documentation
- "Example: Round-Robin Approach to Choosing Inputs" in the SimEvents getting started documentation
- "Constructs Involving Queues and Servers" in the SimEvents getting started documentation
- "Example of a Logical Queue" in the SimEvents getting started documentation
- "Example: Waiting Time in LIFO Queue" in the SimEvents user guide documentation

See Also

LIFO Queue, Priority Queue

"Basic Queues and Servers" in the SimEvents getting started documentation

Get Attribute

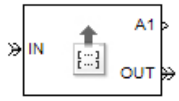
Purpose

Output value of entity's attribute

Library

Attributes

Description



This block outputs signals using data from attributes of entities. For each arriving entity, the block updates the signal at the signal output ports using values of the attributes named in the block dialog box. The block also outputs the entity unchanged.

Manipulating the Rows of the Table

Each attribute corresponds to a row in the table on the **Get Attribute** tab in the block's dialog box. Buttons to the left of the table let you manipulate rows in the table:

- To add a new row to the table, click the Add button . By default, the attribute name in the new row is unique within the table.
- To duplicate a row, select it and click the Copy button .
- To remove a row, select it and click the Delete button .

Note The dialog box does not ask you to confirm the deletion and does not offer an undo operation. However, if you delete a row by mistake, you can click **Cancel** to ignore unapplied changes.

Note Deleting a row and applying the change might affect signal output ports corresponding to other rows of the table. For example, if the block has a signal output port **A2** and you delete the row marked **A1**, then the block renames **A2** as **A1**. Check that any signal that connects to the renamed port is still connected as you expect.

Within each row, you can specify aspects of the block's behavior that relate to the attribute that corresponds to that row.

Missing Attributes

You can specify the block's behavior if the arriving entity does not possess an attribute listed in the table of the block dialog box. Use the **When Attribute Is Missing** parameter in the table row for that attribute, as described below.

Parameter Value	Block Behavior in Case of Missing Attribute
`Error`	The block issues an error message and halts the simulation. In this case, the **Default Value** and **Treat Vector as 1-D** parameters in the table row for that attribute are disabled.
`Default value`	The block outputs a default value that you specify using the **Default Value** and **Treat Vector as 1-D** parameters in the table row for that attribute. The simulation proceeds.
`Warn`	The block outputs a default value that you specify using the **Default Value** and **Treat Vector as 1-D** parameters in the table row for that attribute. The block also issues a warning in the MATLAB Command Window. The simulation proceeds.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Entity Output Ports

Label	Description
OUT	Port for departing entities.

Signal Output Ports

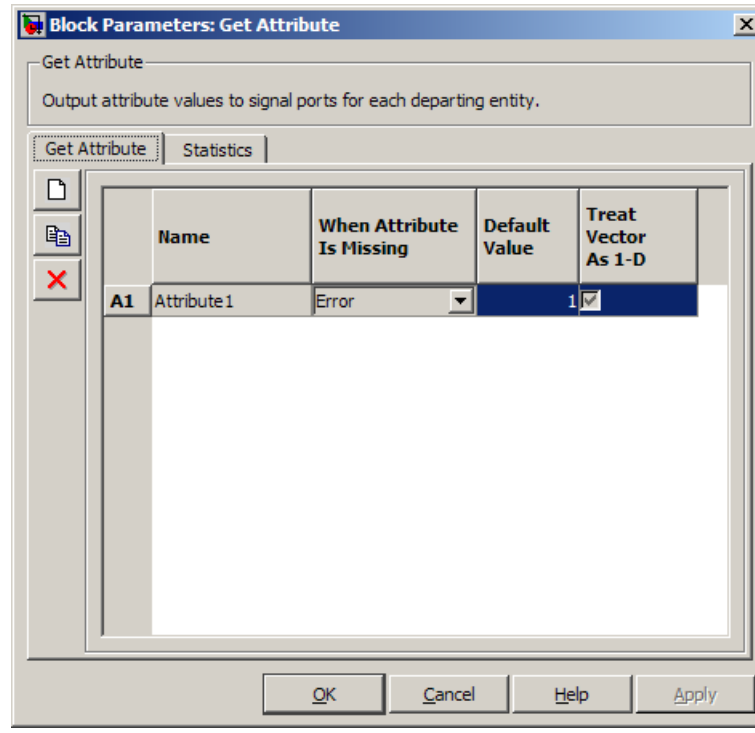
Label	Description	Time of Update When Statistic Is On	Order of Update
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	2
A*x*, where ***x*** = 1, 2, 3,...	Value of the attribute specified in the **A*x*** row of the table in the dialog.	After entity departure	1

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

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

Dialog Box

Get Attribute Tab



Name

The name of the attribute to query.

When Attribute Is Missing

The response of the block when the entity does not possess an attribute named in the table.

Default Value

The value for the corresponding output signal if the entity does not possess an attribute named in the table. See “Attribute Value Support” in the SimEvents user guide documentation to learn what kind of data you can use as a default value. This field

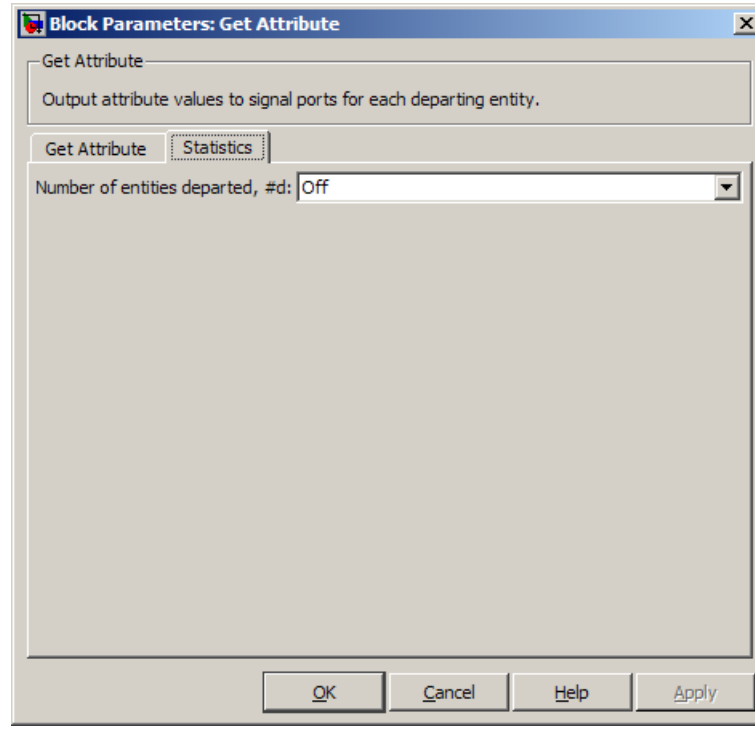
appears only if you set **When Attribute Is Missing** to `Default value` or `Warn`.

Treat Vector as 1-D

This option affects attributes whose **When Attribute Is Missing** parameter is set to `Default value` or `Warn`, and whose **Default Value** parameter evaluates to an N-element row or column vector. Selecting this option causes the block to consider the default value as a vector of length N. Otherwise, the block considers the default value as a multidimensional array.

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
Controls the presence and behavior of the signal output port labeled **#d**.

Examples

- "Adding Event-Based Behavior" in the SimEvents getting started documentation
- "Using Block Diagrams to Manipulate Attributes" in the SimEvents user guide documentation

See Also

Set Attribute

"Accessing Attributes of Entities" in the SimEvents user guide documentation

Get Attribute (Obsolete)

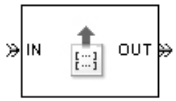
Purpose

Output value of entity's attribute

Library

Attributes

Description



Note The Get Attribute block in the `simeventsattributes1` library will be removed in a future release. Use the Get Attribute block in the `simeventsattributes2` library instead.

This block outputs signals from up to four attributes. For each entity, the block updates the signal at the **A1**, **A2**, **A3**, and/or **A4** signal output port using the value of the attribute named in the corresponding tab of the dialog box. The block also outputs the entity unchanged.

To query fewer than the maximum number of attributes, you can deactivate unused tabs by setting **Send attribute value to signal port A4**, for example, to `Off`.

To specify the names of attributes you want to query, use the **Attribute name** parameter in the dialog box.

Missing Attributes

If the block is configured to retrieve the value of an attribute that an arriving entity does not possess, then the block can react in one of these ways:

- Issue an error message and halt the simulation.
- Output a default value that you specify. The simulation proceeds.
- Output a default value that you specify and also issue a warning in the MATLAB Command Window. The simulation proceeds.

You determine the reaction using the **Action for missing attribute** parameter on each attribute tab of the dialog. The error and warning options can be useful for debugging a simulation (for example, to locate a mistyped attribute name in a dialog box).

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	2
A*x*, where ***x*** = 1, 2, 3, 4	Value of the attribute specified on the **A*x*** tab of the dialog. This port appears only if you set **Send attribute value to signal port A*x*** on the **A*x*** tab to `On`.	After entity departure	1

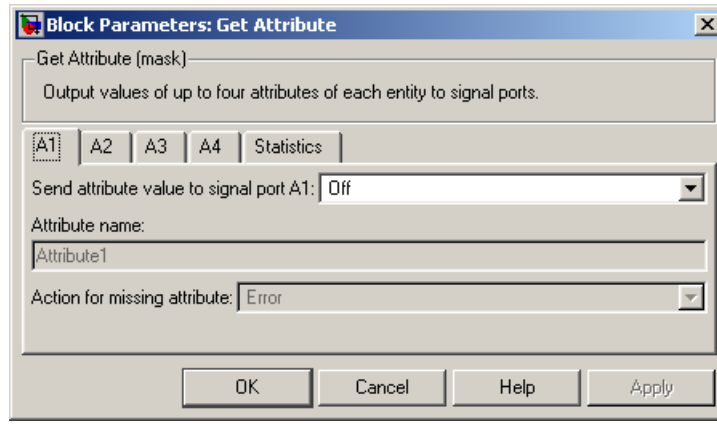
Output signals having the same number in the Order of Update column in the table above are updated in an arbitrary sequence relative to each other; you should not rely on a specific sequence for your simulation results.

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

Dialog Box

A1, A2, A3, A4 Tabs

The **A1**, **A2**, **A3**, and **A4** tabs have similar parameter choices. By assigning different values to the parameters, you can configure this block to retrieve the values of up to four different attributes for each entity that the block processes.



Send attribute value to signal port A*x*, where ***x*** = 1, 2, 3, 4
: Indicates whether the block creates an output signal with the value of an attribute. Choosing `Off` indicates that you are not using this tab of the dialog and makes the parameters below inactive or invisible.

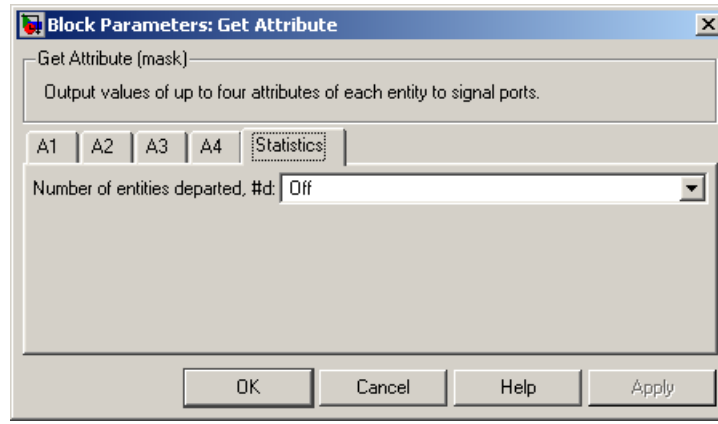
Attribute name
: The name of the attribute to query.

Action for missing attribute
: The response of the block when the entity does not possess the attribute named above.

Default Value
: The value for the corresponding output signal if the entity does not possess the attribute named above. This field appears only if you set **Action for missing attribute** to `Output default value` or `Output default value and warn`.

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
: Controls the presence and behavior of the signal output port labeled **#d**.

Examples

- "Adding Event-Based Behavior" in the SimEvents getting started documentation
- "Using Block Diagrams to Manipulate Attributes" in the SimEvents user guide documentation

See Also

Get Attribute, Set Attribute

"Accessing Attributes of Entities" in the SimEvents user guide documentation

Infinite Server

Purpose

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; see "Forcing Departures Using Timeouts" in the SimEvents user guide documentation for details about timeouts.

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. For common problems and troubleshooting tips, see "Unexpected Use of Old Value of Signal" in the SimEvents user guide documentation.

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 port appears 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. This port appears 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

Signal Output Ports (Continued)

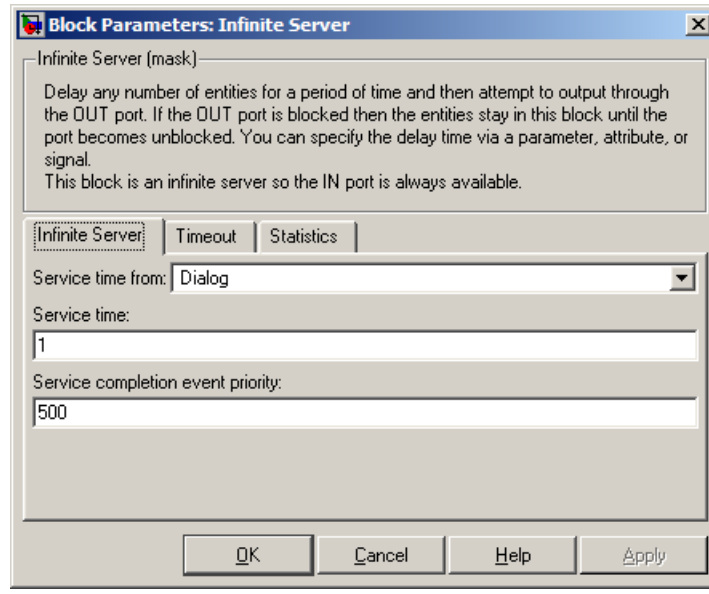
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#n	Number of entities in the block.	After entity arrival and after entity departure	4
pe	The value is 1 if the block is holding any pending entities, that is, entities that have completed their service but cannot depart because the **OUT** port is blocked. The value of this signal is 0 if the block has no pending entities or if an entity departure is imminent.	After service completion and when entity departure is imminent	1
#pe	The number of entities that have completed their service but cannot depart because the **OUT** port is blocked.	After service completion (increase in value) and after entity departure (decrease in value)	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.

Dialog Box

Infinite Server Tab



Service time from

Determines whether the service time is computed from a parameter in this dialog box, a signal input port, or an attribute of the entity being served.

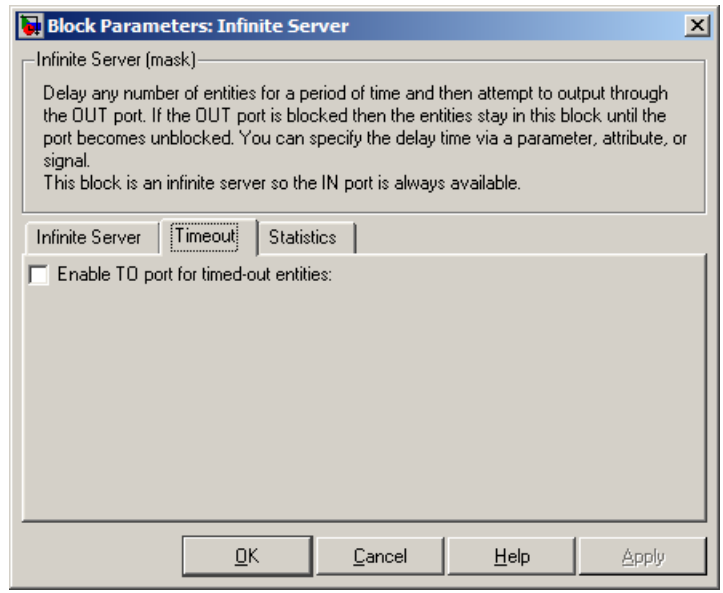
Service time

The service time, in seconds, for all entities. This field appears 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. This field appears 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.

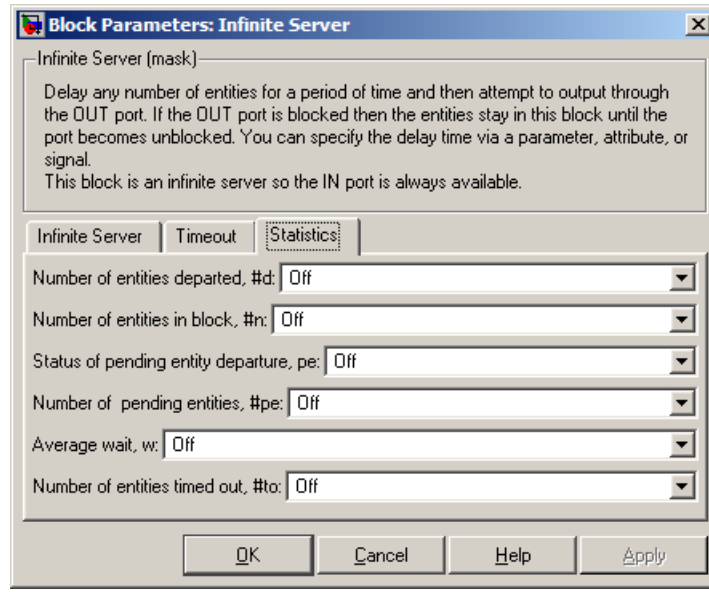
Timeout Tab



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 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
: Controls the presence and behavior of the signal output port labeled **#d**.

Number of entities in block
: Controls the presence and behavior of the signal output port labeled **#n**.

Status of pending entity departure
: Controls the presence of the signal output port labeled **pe**.

Number of pending entities
Controls the presence and behavior of the signal output port labeled **#pe**.

Average wait
Controls the presence and behavior of the signal output port labeled **w**.

Number of entities timed out
Controls the presence and behavior of the signal output port labeled **#to**.

Examples

- "Adding Event-Based Behavior" in the SimEvents getting started documentation
- "Restarting a Timer from Zero" in the SimEvents user guide documentation
- "Example: Counting Simultaneous Departures from a Server" in the SimEvents user guide documentation

See Also

Single Server, N-Server

"Basic Queues and Servers" in the SimEvents getting started documentation

Purpose

Output specified value until first sample time hit

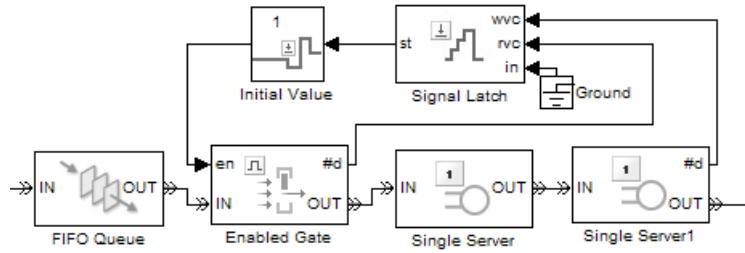
Library

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

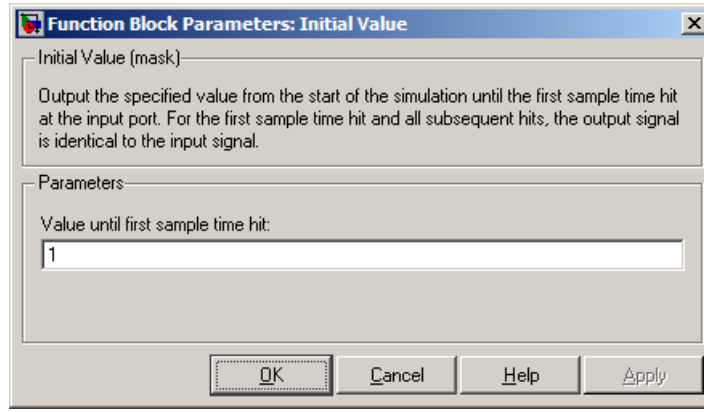
Label	Description
None	The first sample time hit in this signal causes the block to stop using the initial value from the block dialog box. From then on, the output signal is identical to the input signal.

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.

Dialog Box

The initial output value, which is in effect strictly before the first sample time hit of the input signal, is the value of the **Value until first sample time hit** parameter.



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.

Examples

- "Example: Controlling Joint Availability of Two Servers" in the SimEvents user guide documentation
- "Example: Failure and Repair of a Server" in the SimEvents user guide documentation
- "Example: Limiting the Time Until Service Completion" in the SimEvents user guide documentation

See Also

IC, Signal Latch

"Specifying Initial Values of Event-Based Signals" in the SimEvents user guide documentation

Input Switch

Purpose

Accept entities from selected entity input port

Library

Routing

Description

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 possible rules the block uses for selecting an entity input port, as well as the corresponding values of the **Switching criterion** parameter in the dialog box, are listed in the table below.

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, then you should include a storage block in each of those paths.

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, you should 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 port appears only if you set **Switching criterion** to `From signal port p`.

Entity Output Ports

Label	Description
OUT	Port for departing entities.

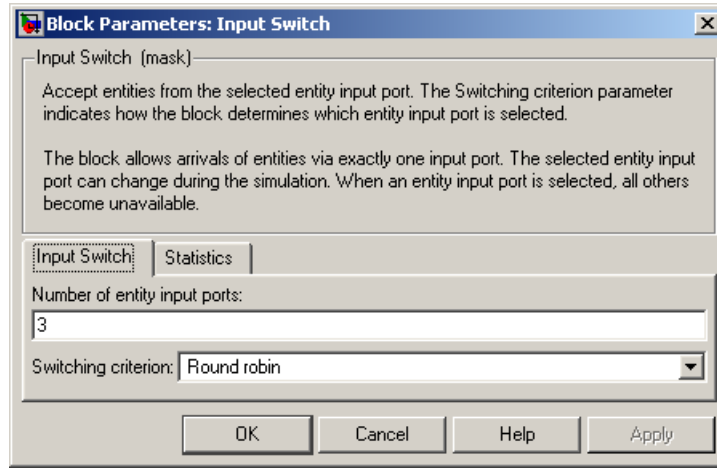
Signal Output Ports

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.

Dialog Box

Input Switch Tab



Number of entity input ports
Determines how many entity input ports the block has. This parameter must be a literal value, not a variable or an expression requiring evaluation.

Switching criterion
The rule that determines which entity input port is selected for receiving entities.

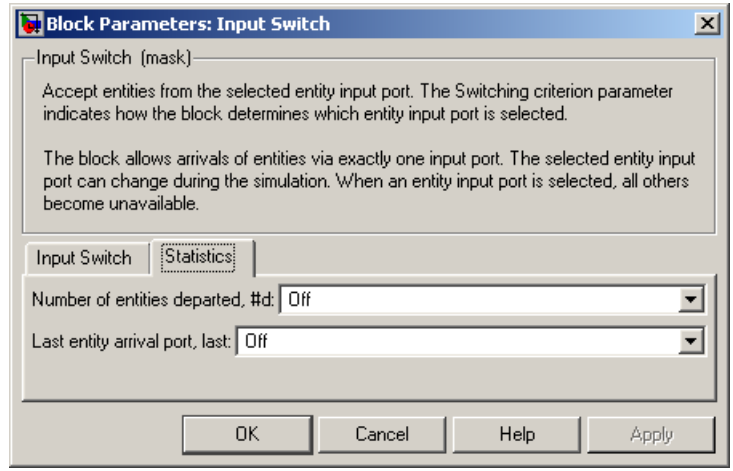
Initial seed
A nonnegative integer that initializes the random number generator used to select an entity input port. This field appears only if you set **Switching criterion** to `Equiprobable`.

Resolve simultaneous signal updates according to event priority
Select this option to explicitly prioritize the port-selection event, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority `SYS1` on the event calendar. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears 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. For details, see "Specifying Event Priorities to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears 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 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
: Controls the presence and behavior of the signal output port labeled **#d**.

Last entity arrival port
: Controls the presence of the signal output port labeled **last**.

Examples

- "Example: Choices of Values for Event Priorities" in the SimEvents user guide documentation
- "Example: Round-Robin Approach to Choosing Inputs" in the SimEvents getting started documentation
- "Example: Compound Switching Logic"

See Also

Output Switch

"Using the Input Switch" in the SimEvents getting started documentation

Purpose

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 block to the Signal Scope 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.

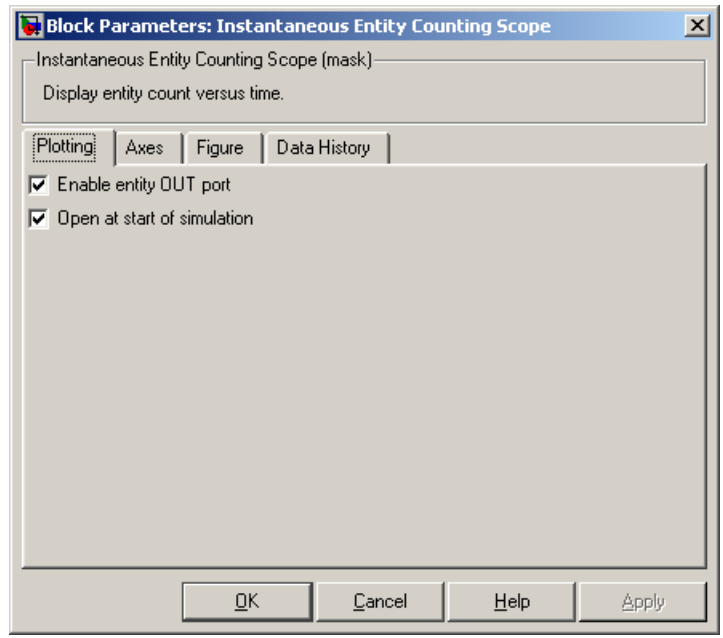
Entity Output Ports

Label	Description
OUT	Port for departing entities. This port appears only if you select **Enable entity OUT port**.

Dialog Box

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

Plotting Tab



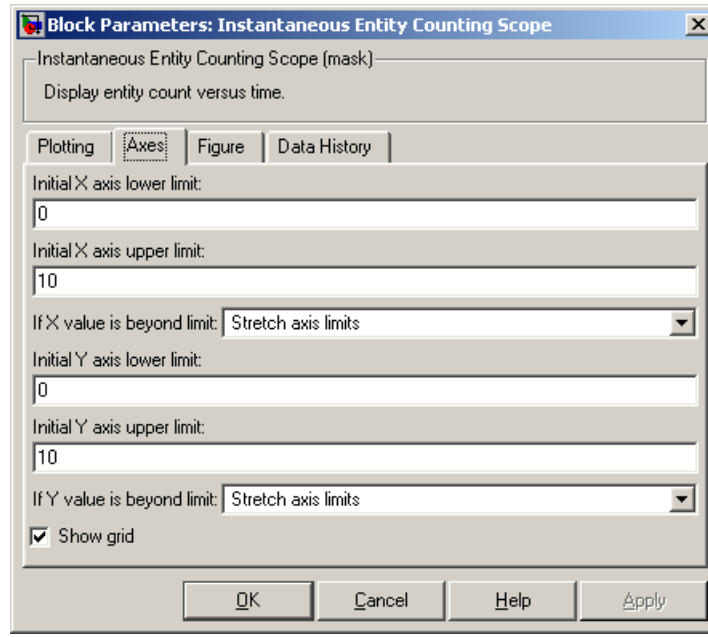
Enable entity OUT port

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

Open scope at start of simulation

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

Axes Tab



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

If X value is beyond limit
Determines how the plot changes if one or more X values are not within the limits shown on the X axis. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

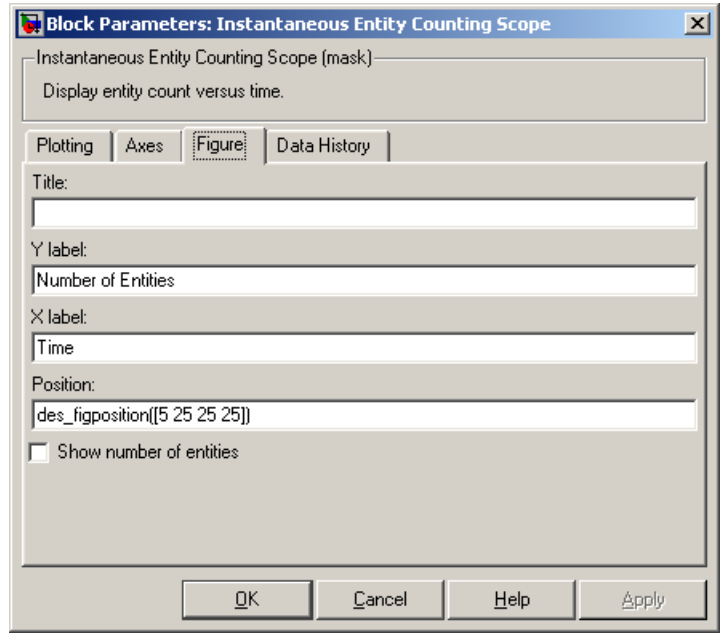
Initial Y axis lower limit, **Initial Y axis upper limit**
The interval shown on the Y axis at the beginning of the simulation. The interval might change from this initial setting

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. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

Show grid
Toggles the grid on and off.

Figure Tab



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

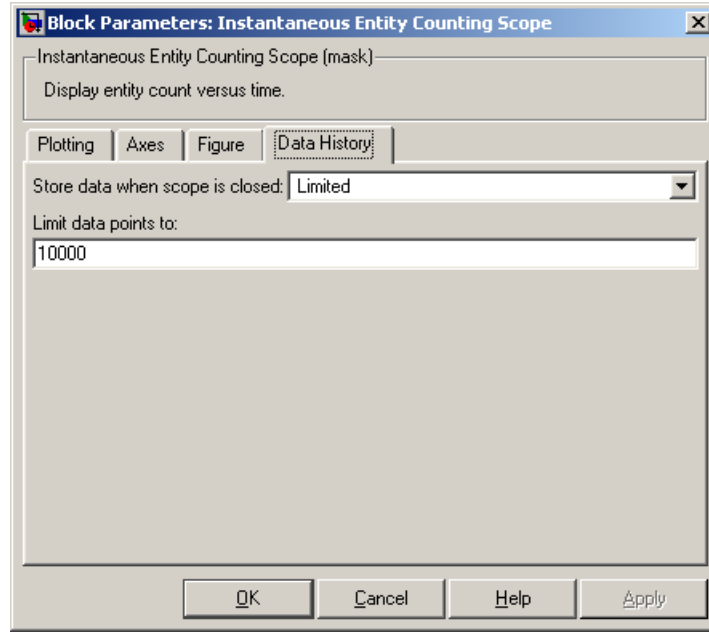
Y label
Text that appears to the left of the vertical axis.

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. This field appears only if you set **Store data when scope is closed** to `Limited`.

Examples

- "Example: Counting Simultaneous Departures from a Server" in the SimEvents user guide documentation
- "Example: Synchronizing Service Start Times with the Clock" in the SimEvents user guide documentation

See Also

Entity Departure Counter, Instantaneous Event Counting Scope

"Plotting Data" in the SimEvents user guide documentation, "Counting Entities" in the SimEvents user guide documentation

Purpose

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 signal's initial condition.

Ports

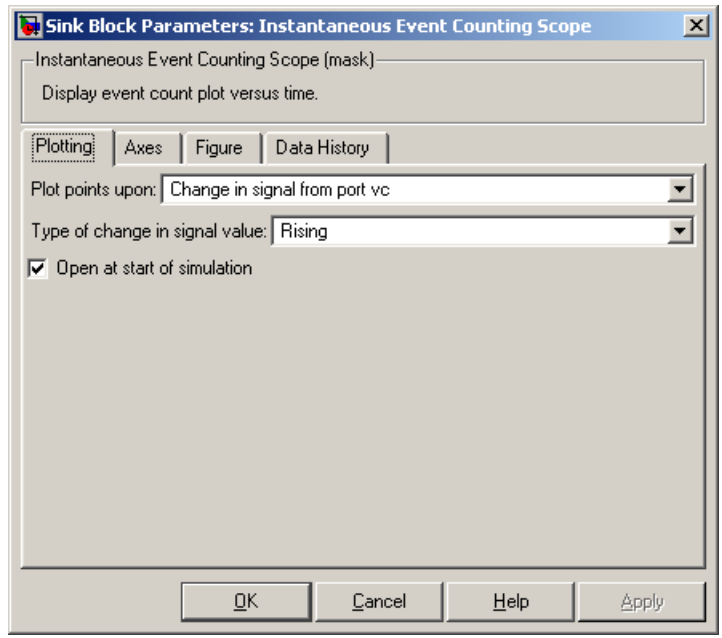
Signal Input Ports

Label	Description
ts	Signal whose updates increment the counter. This port appears only if you set **Plot points upon** to `Sample time hit from port ts`.
tr	Trigger signal whose edges increment the counter. This port appears only if you set **Plot points upon** to `Trigger from port tr`.
vc	Signal whose numerical changes in value increment the counter. This port appears only if you set **Plot points upon** to `Change in signal from port vc`.
fcn	Function-call signal that indicates when to increment the counter. This port appears only if you set **Plot points upon** to `Function call from port fcn`.

Dialog Box

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

Plotting Tab



Plot points upon

The type of event that indicates when the block increments its counter.

Trigger type

Determines whether rising, falling, or either type of trigger edge causes the block to increment its counter. This field appears only if you set **Plot points upon** to `Trigger from port tr`.

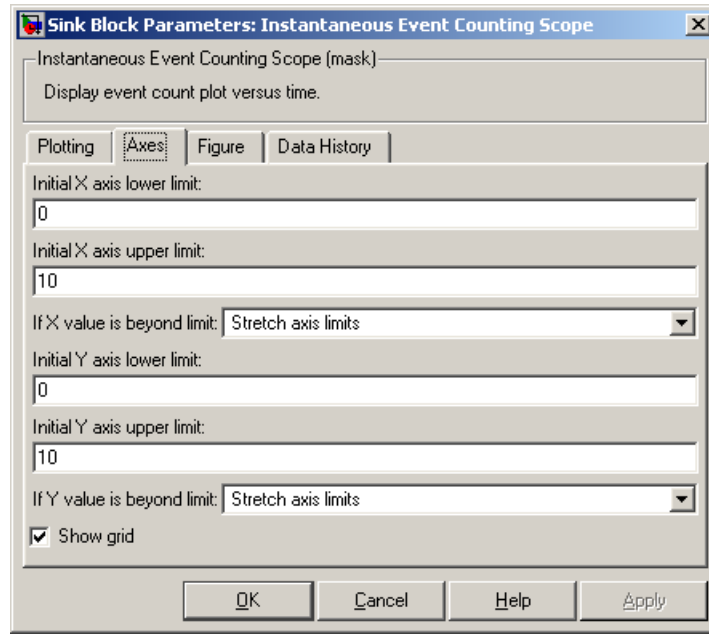
Type of change in signal value

Determines whether rising, falling, or either type of value change causes the block to increment its counter. This field appears 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.

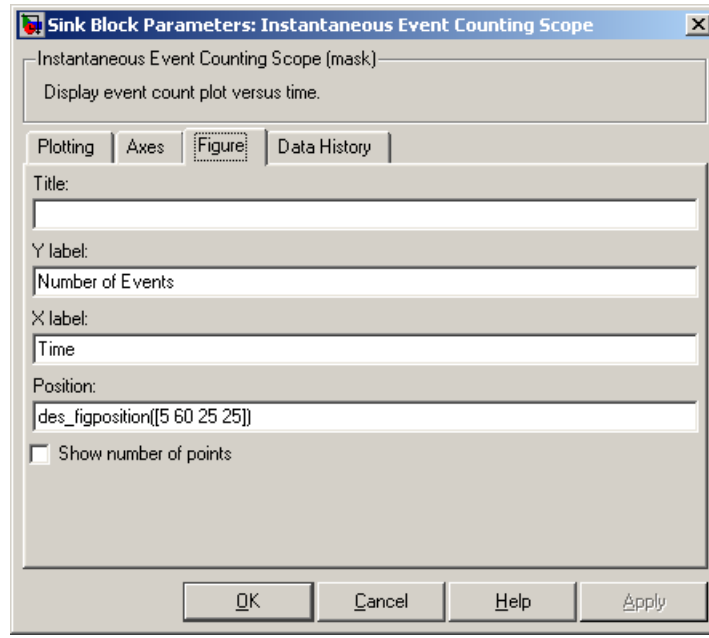
If X value is beyond limit
Determines how the plot changes if one or more X values are not within the limits shown on the X axis. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

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. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

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.

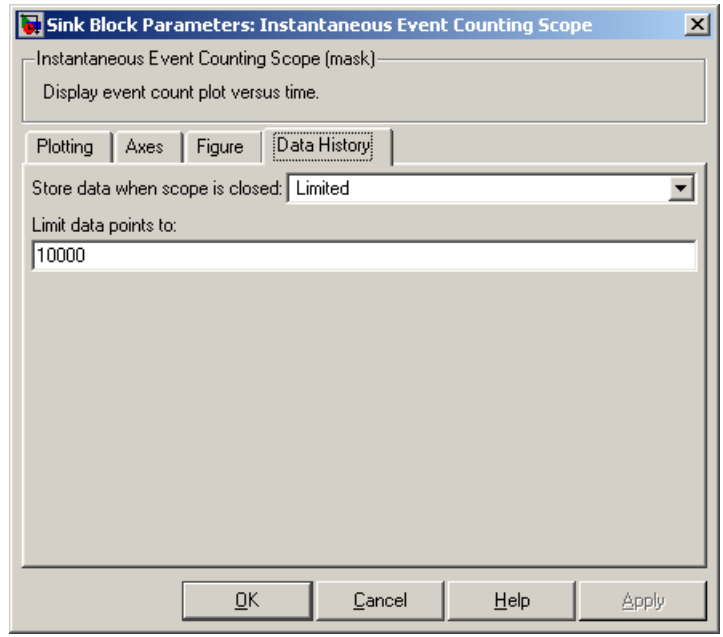
X label
Text that appears below the horizontal axis.

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. This field appears only if you set **Store data when scope is closed** to `Limited`.

Examples

See “Example: Plotting Event Counts to Check for Simultaneity” in the SimEvents user guide documentation.

See Also

Signal Scope, Instantaneous Entity Counting Scope

“Plotting Data” in the SimEvents user guide documentation, “Observing Events” in the SimEvents user guide documentation

LIFO Queue

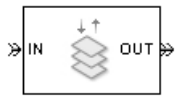
Purpose

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 and N>0. 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 N>0 and this block stores exactly N entities. In this case, the queue is said to be full.

If N=0, then the availability of the **IN** port matches the availability of the subsequent block's entity input port to which this block's **OUT** port is connected. If N=0, all arriving entities depart immediately because this block cannot store entities.

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. This port appears 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

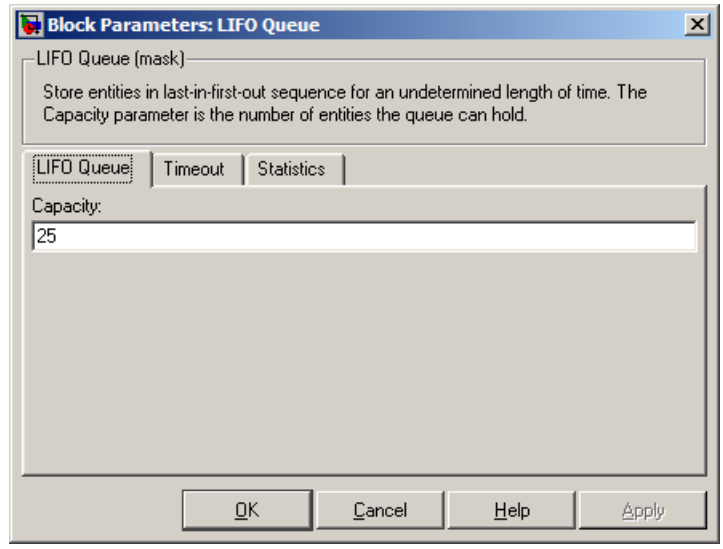
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block via the **OUT** port since the start of the simulation.	After entity departure via the **OUT** port	4
#n	Number of entities currently in the queue.	After entity arrival and after entity departure	3
pe	A value of 1 indicates when the block tries and fails to output an entity. A value of 0 indicates when the queue is empty or an entity departure is imminent.	After entity arrival and when entity departure is imminent	1
w	Sample mean of the waiting times in this block for all entities that have departed via any port.	After entity departure	2
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. If the entity arrives at an empty queue and immediately departs, then **len** has one sample time hit, not two.	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.

Dialog Box

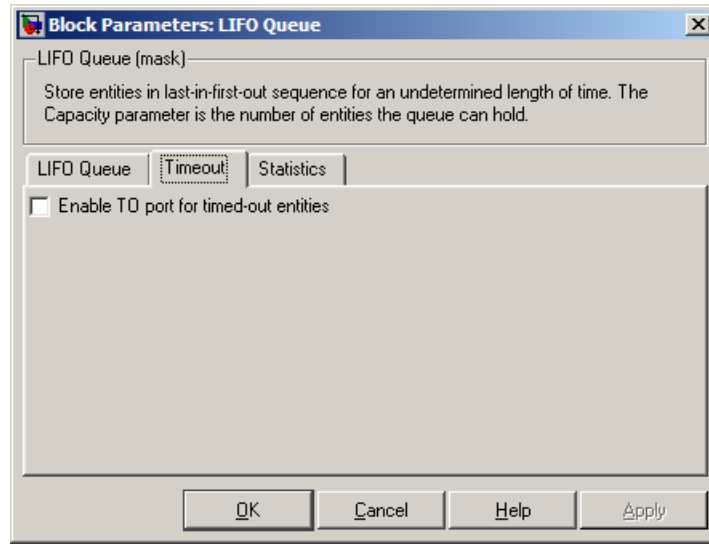
LIFO Queue Tab



Capacity

Determines how many entities the block can store at a time.

Timeout Tab

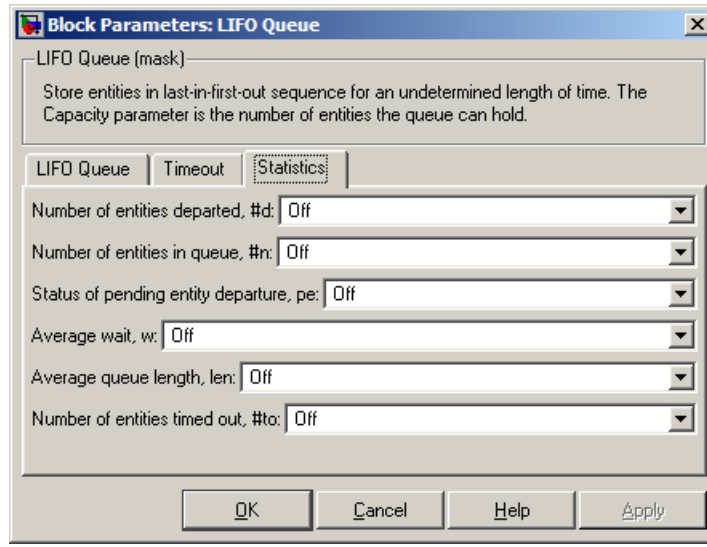


Enable TO port for timed-out entities

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 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

Controls the presence and behavior of the signal output port labeled **#d**.

Number of entities in queue

Controls the presence and behavior of the signal output port labeled **#n**.

Status of pending entity departure

Controls the presence of the signal output port labeled **pe**.

Average wait

Controls the presence and behavior of the signal output port labeled **w**.

Average queue length

Controls the presence and behavior of the signal output port labeled **len**.

Number of entities timed out
Controls the presence and behavior of the signal output port labeled **#to**.

Examples

See “Example: Waiting Time in LIFO Queue” in the SimEvents user guide documentation.

See Also

FIFO Queue, Priority Queue

“Using a LIFO Queuing Discipline” in the SimEvents user guide documentation

N-Server

Purpose

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; see "Forcing Departures Using Timeouts" in the SimEvents user guide documentation for details about timeouts.

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. For common problems and troubleshooting tips, see "Unexpected Use of Old Value of Signal" in the SimEvents user guide documentation.

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

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 port appears 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. This port appears 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

Signal Output Ports (Continued)

Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#n	Number of entities currently in the block, between 0 and N.	After entity arrival and after entity departure	4
pe	The value is 1 if the block is holding any pending entities, that is, entities that have completed their service but cannot depart because the **OUT** port is blocked. The value of this signal is 0 if the block has no pending entities or if an entity departure is imminent.	After service completion and when entity departure is imminent	1
#pe	The number of entities that have completed their service but cannot depart because the **OUT** port is blocked.	After service completion (increase in value) and after entity departure (decrease in value)	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

Signal Output Ports (Continued)

Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
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 suppress signal updates until specific occurrences cause updates. In `On` mode, updates occur after an entity departure and after an entity arrival. In `Upon stop or pause` mode, stopping or pausing the simulation causes an update.	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.

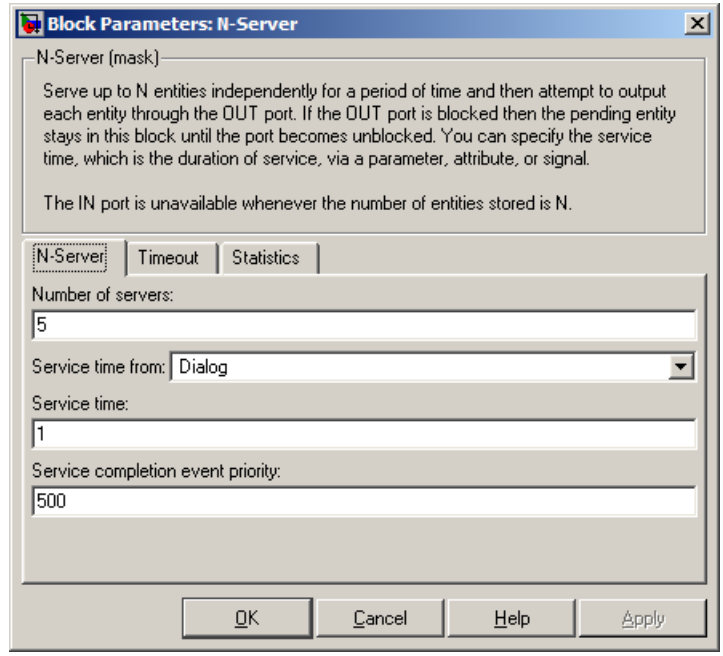
A more precise definition of the utilization signal **util** at an update time $T>0$ is

$$\frac{1}{T}\sum_{k}\left(\frac{(\#n)_k}{N}\right)\cdot \text{length}(I_k)$$

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 $\#n/N$.

Dialog Box

N-Server Tab



Number of servers

The number of servers the block represents, N.

Service time from

Determines whether the service time is computed from a parameter in this dialog box, an input signal, or an attribute of the entity being served.

Service time

The service time, in seconds, for all entities. This field appears only if you set **Service time from** to `Dialog`.

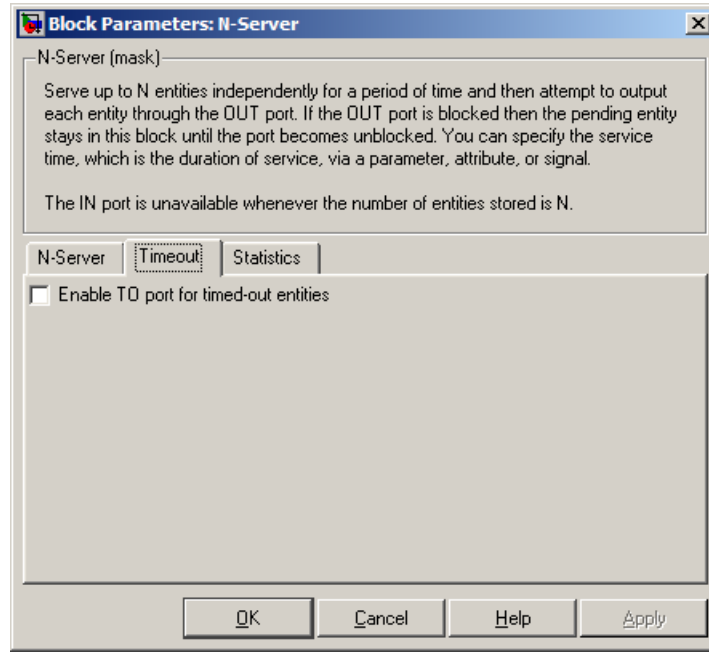
Attribute name

The name of the attribute whose value the block uses as the service time for an entity. This field appears 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

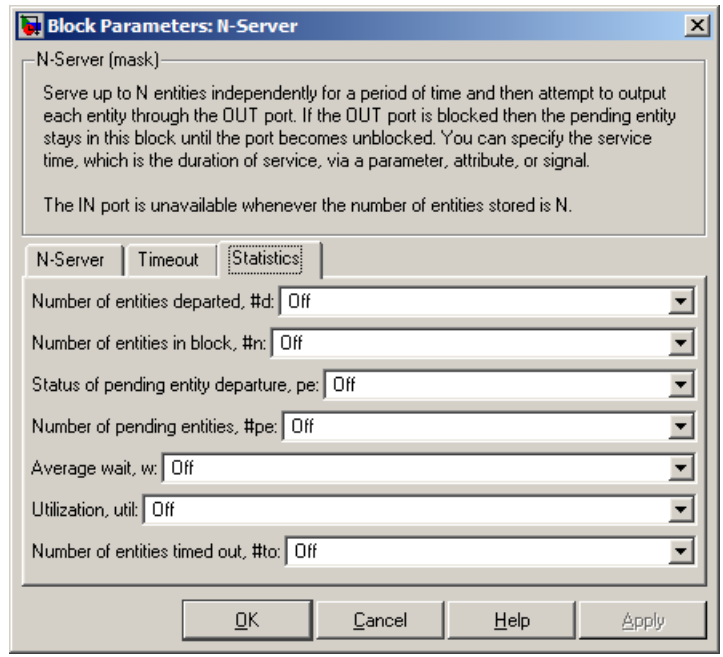


Enable TO port for timed-out entities

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 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
Controls the presence and behavior of the signal output port labeled **#d**.

Number of entities in block
Controls the presence and behavior of the signal output port labeled **#n**.

Status of pending entity departure
Controls the presence of the signal output port labeled **pe**.

Number of pending entities
Controls the presence and behavior of the signal output port labeled **#pe**.

Average wait
Controls the presence and behavior of the signal output port labeled **w**.

Utilization
Controls the presence and behavior of the signal output port labeled **util**.

Number of entities timed out
Controls the presence and behavior of the signal output port labeled **#to**.

Examples

See "Example: M/M/5 Queuing System" in the SimEvents user guide documentation.

See Also

Single Server, Infinite Server

"Modeling Multiple Servers" in the SimEvents user guide documentation

Output Switch

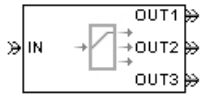
Purpose

Select entity output port for departure

Library

Routing

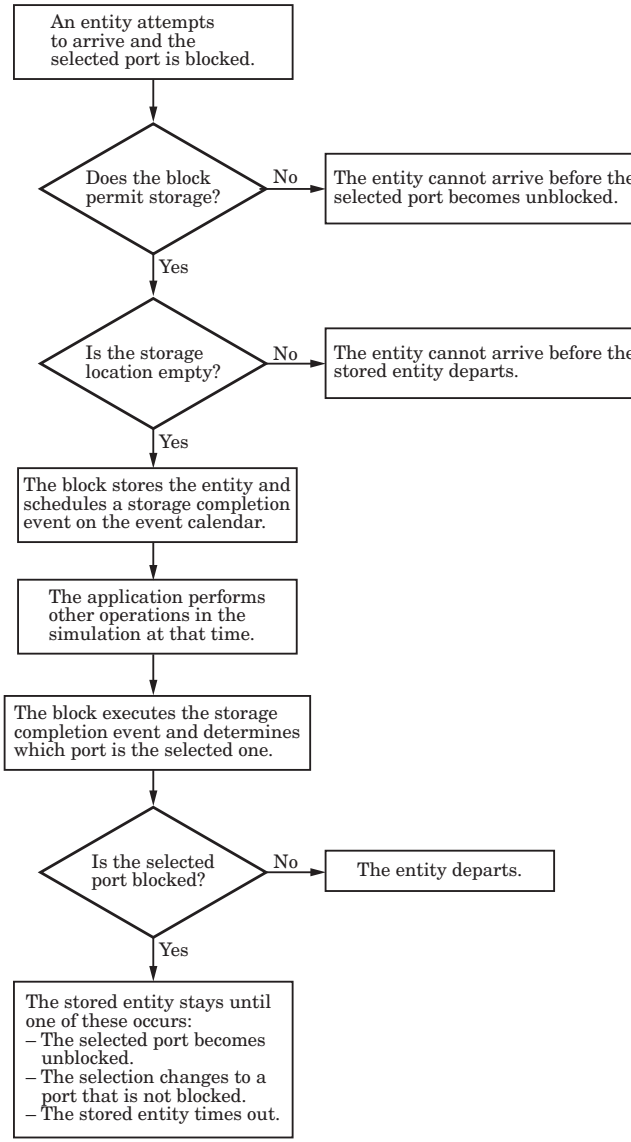
Description



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.
Does the block permit storage?
No
The entity cannot arrive before the selected port becomes unblocked.
Yes
Is the storage location empty?
No
The entity cannot arrive before the stored entity departs.
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.
Is the selected port blocked?
No
The entity departs.
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. For details on timeouts, see "Forcing Departures Using Timeouts" in the SimEvents user guide documentation.

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.

Switching criterion Value	Description
`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 "Output Switching Based on a Signal" in the SimEvents user guide documentation. For common problems and troubleshooting tips, see "Unexpected Use of Old Value of Signal" in the SimEvents user guide documentation.

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 port appears 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. This port appears 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.

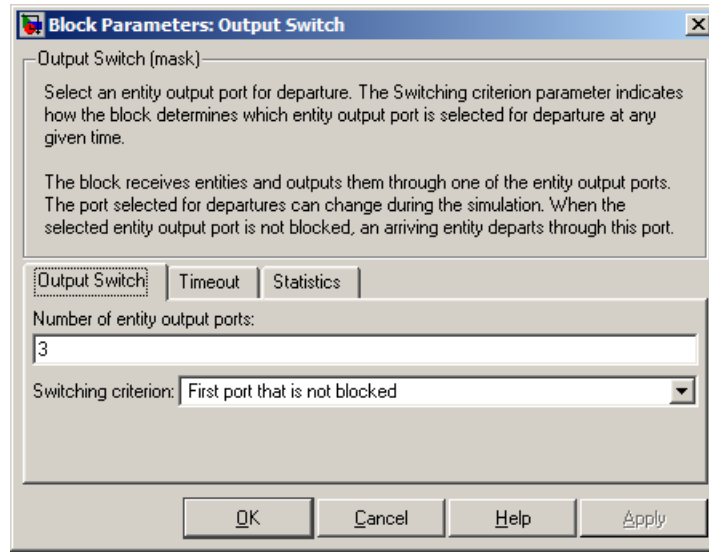
Signal Output Ports

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 when the block stores an entity after the entity tries and fails to depart. A value of 0 indicates when the storage location is empty or an entity departure is imminent.	After storage completion event if the entity fails to depart, and when entity departure is imminent	1
last	Index of the output port through which the last entity departed, excluding timed-out entities. Aside from the initial condition, 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.

Dialog Box

Output Switch Tab



Number of entity output ports

Determines how many entity output ports the block has. This parameter must be a literal value, not a variable or an expression requiring evaluation.

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. This field appears 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 “Specifying an Initial Port Selection” in the

SimEvents user guide documentation. This field appears 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. This field appears 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. For details, see "Using the Storage Option to Prevent Latency Problems" in the SimEvents user guide documentation. This field appears only if you set **Switching criterion** to `From signal port p`.

Resolve simultaneous signal updates according to event priority

Select this option to explicitly prioritize the port-selection event, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority `SYS1` on the event calendar. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears only if you set **Switching criterion** to `From signal port p` and do not select **Store entity before switching**.

Event priority

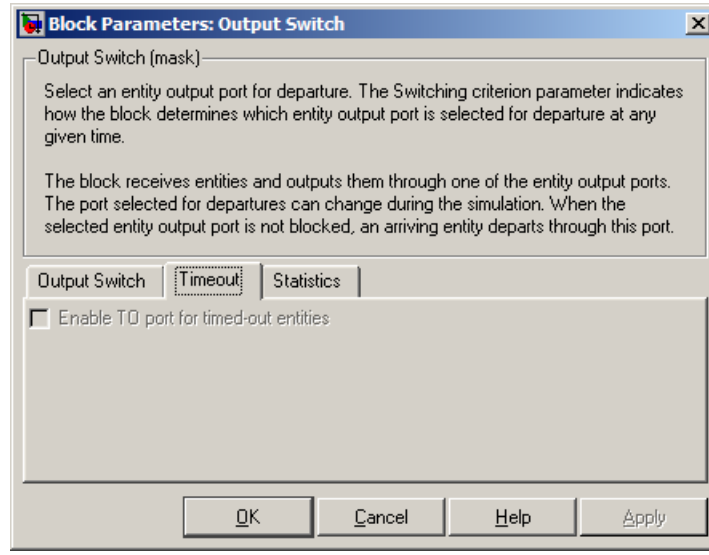
The priority of the port-selection event, relative to other simultaneous events in the simulation. For details, see "Specifying Event Priorities to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears only if you set **Switching criterion** to `From signal 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. This field appears only if you set **Switching criterion** to `From attribute`.

Timeout Tab

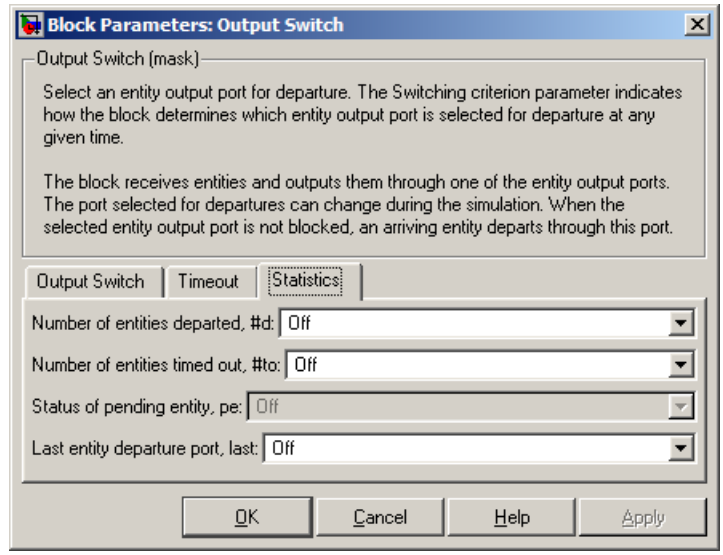


Enable TO port for timed-out entities

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 block reference page.

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
: Controls the presence and behavior of the signal output port labeled **#d**.

Number of entities timed out
: Controls the presence of the signal output port labeled **#to**.

Status of pending entity
: Controls the presence of the signal output port labeled **pe**. This parameter can have the value `On` 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
Controls the presence of the signal output port labeled **last**.

Examples

- "Example: Selecting the First Available Server" in the SimEvents getting started documentation
- "Example: Using an Attribute to Select an Output Port" in the SimEvents getting started documentation
- "Example: A Packet Switch" in the SimEvents getting started documentation
- "Example: Choosing the Shortest Queue" in the SimEvents user guide documentation
- "Example: Using Servers in Shifts" in the SimEvents user guide documentation

See Also

Input Switch

"Using the Output Switch" in the SimEvents getting started documentation, "Output Switching Based on a Signal" in the SimEvents user guide documentation

Path Combiner

Purpose

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. For more information, see "Managing Simultaneous Events" in the SimEvents user guide documentation. For an example, see the "No blockage" case in "Connecting Multiple Queues to the Output Switch" in the SimEvents getting started documentation and note the dependence on generation event priority values. 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.

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. For an example illustrating when this parameter is significant, see "Combining Entity Paths" in the SimEvents getting started documentation.

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**.
`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 port appears only if you set **Input port precedence** to `From signal port p`.

Entity Output Ports

Label	Description
OUT	Port for departing entities.

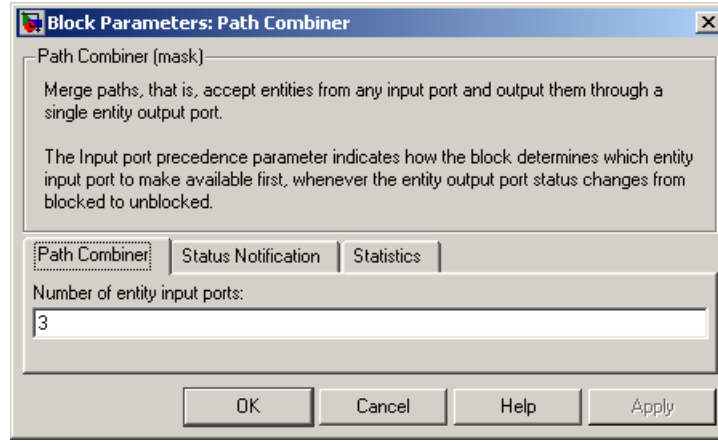
Signal Output Ports

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.

Dialog Box

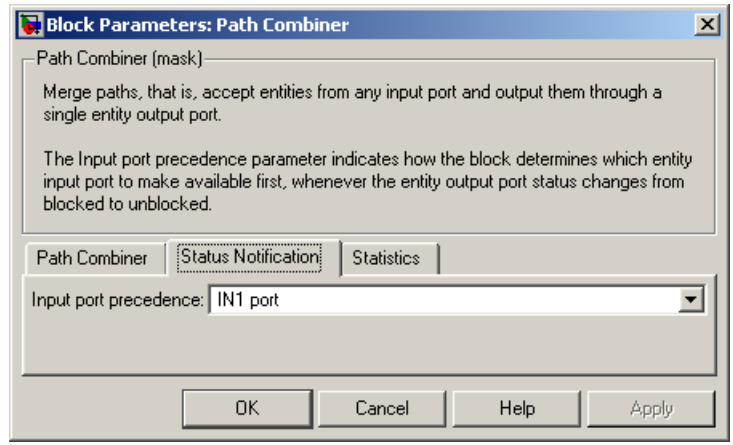
Path Combiner Tab



Number of entity input ports

Determines how many entity input ports the block has. This parameter must be a literal value, not a variable or an expression requiring evaluation.

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. This field appears only if you set **Input port precedence** to `Equiprobable`.

Resolve simultaneous signal updates according to event priority

Select this option to explicitly prioritize the event that updates the port precedence, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority `SYS1` on the event calendar. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents

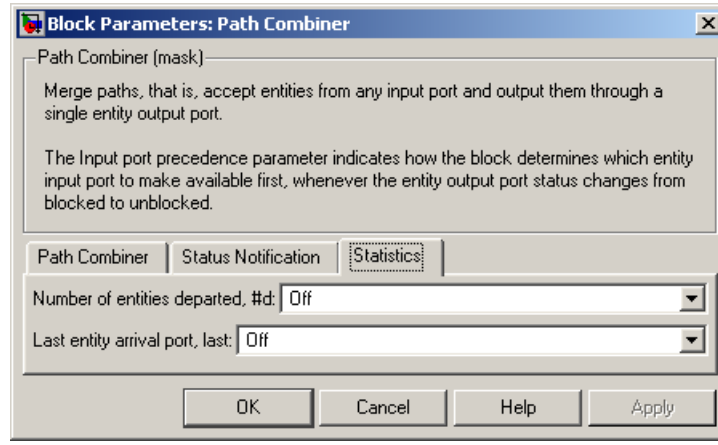
user guide documentation. This field appears only if you set **Switching criterion** to `From signal port p`.

Event priority

The priority of the event that updates the port precedence, relative to other simultaneous events in the simulation. For details, see "Specifying Event Priorities to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears 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 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

Controls the presence and behavior of the signal output port labeled **#d**.

Last entity arrival port
Controls the presence and behavior of the signal output port labeled **last**.

Examples

- "Combining Entity Paths" in the SimEvents getting started documentation
- "Example: A Packet Switch" in the SimEvents getting started documentation
- "Example: First Entity as a Special Case" in the SimEvents user guide documentation

See Also

Input Switch, Output Switch

"Combining Entity Paths" in the SimEvents getting started documentation

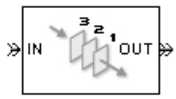
Purpose

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 and N>0. 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 N>0 and 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.

If N=0, then the availability of the **IN** port matches the availability of the subsequent block's entity input port to which this block's **OUT** port is connected. If N=0, all arriving entities depart immediately because this block cannot store entities.

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. This port appears 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 queue.	After entity arrival and after entity departure	3
pe	A value of 1 indicates when the block tries and fails to output an entity. A value of 0 indicates when the queue is empty or an entity departure is imminent.	After entity arrival and when entity departure is imminent	1
w	Sample mean of the waiting times in this block for all entities that have departed via any port.	After entity departure	2

Signal Output Ports (Continued)

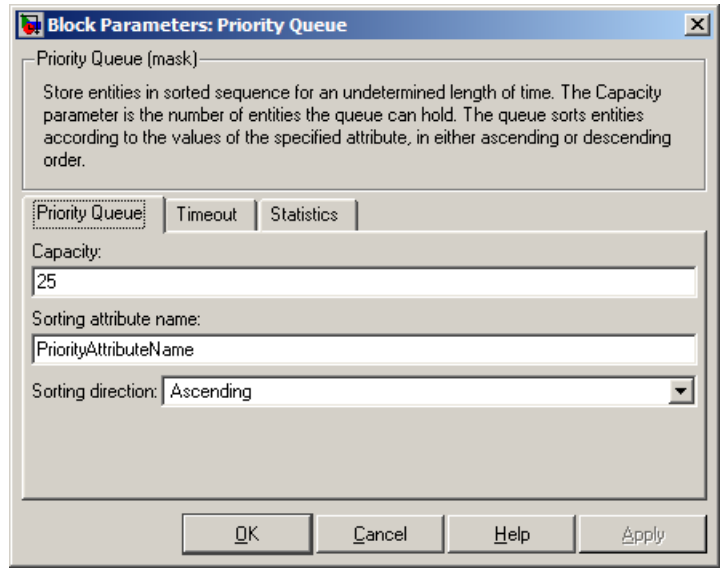
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
len	Average number of entities in the queue over time, that is, the time average of the **#n** signal.	After entity arrival and after entity departure. If the entity arrives at an empty queue and immediately departs, then **len** has one sample time hit, not two.	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.

Dialog Box

Priority Queue Tab



Capacity

Determines how many entities the block can store at a time.

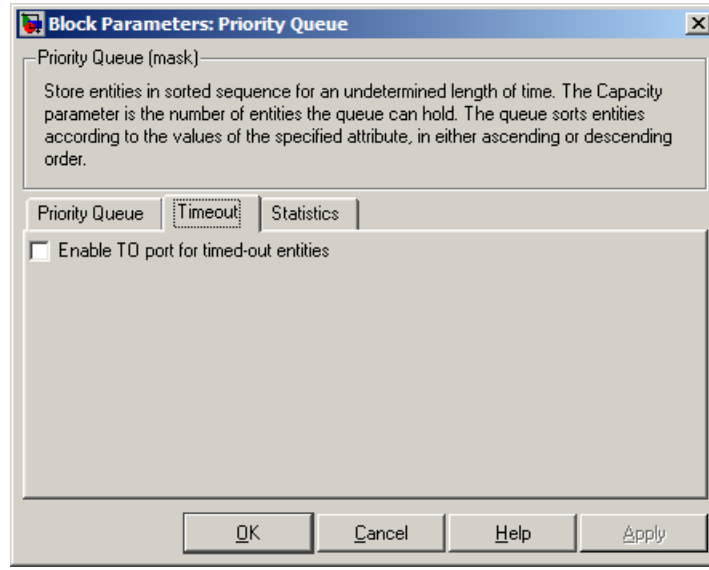
Sorting attribute name

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

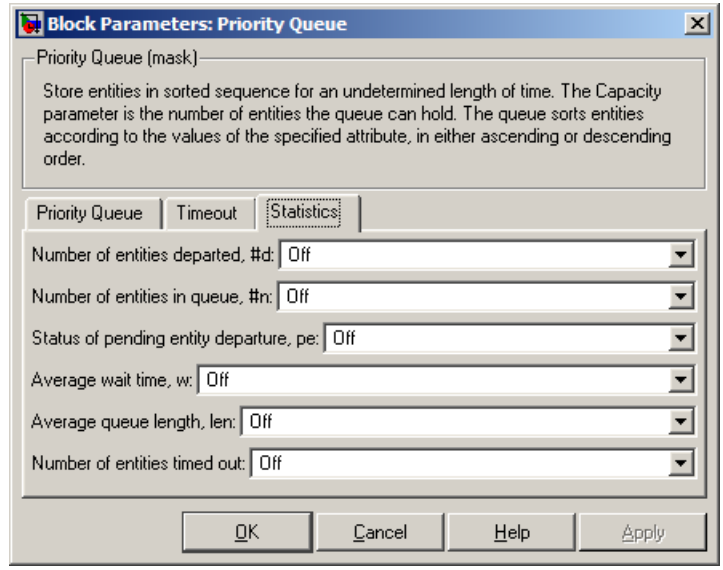


Enable TO port for timed-out entities

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 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

Controls the presence and behavior of the signal output port labeled **#d**.

Number of entities in queue

Controls the presence and behavior of the signal output port labeled **#n**.

Status of pending entity departure

Controls the presence of the signal output port labeled **pe**.

Average wait

Controls the presence and behavior of the signal output port labeled **w**.

Average queue length

Controls the presence and behavior of the signal output port labeled **len**.

Number of entities timed out
Controls the presence and behavior of the signal output port labeled **#to**.

Examples

- "Example: Serving Preferred Customers First" in the SimEvents user guide documentation
- "Example: Preemption by High-Priority Entities" in the SimEvents user guide documentation
- "Example: Rerouting Timed-Out Entities to Expedite Handling" in the SimEvents user guide documentation

See Also

FIFO Queue, LIFO Queue, Single Server

"Sorting by Priority" in the SimEvents user guide documentation

Read Timer

Purpose

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 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.

For more information about using this block with the Start Timer block, see "Using Timers" in the SimEvents user guide documentation.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Entity Output Ports

Label	Description
OUT	Port for departing entities.

Signal Output Ports

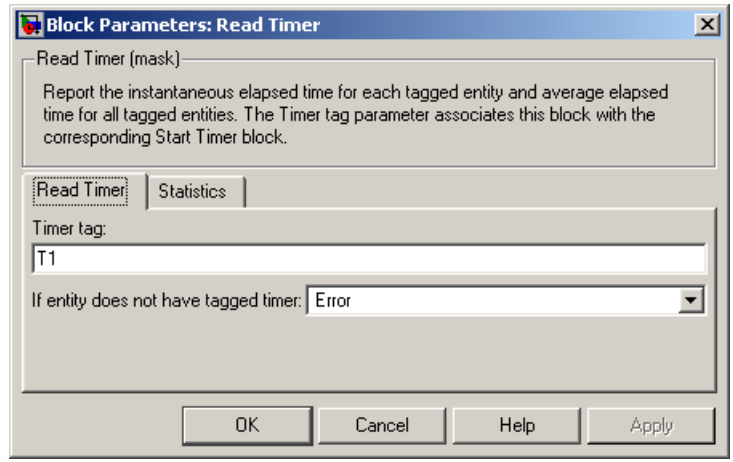
Label	Description	Time of Update When Statistic Is On	Order of Update
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	3
#t	Total number of entities that have departed from this block and possessed a timer of the specified name.	After 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.

Dialog Box

Read Timer Tab



Timer tag

Name of the timer to read. This timer tag corresponds to the **Timer tag** parameter of a Start Timer block in the model.

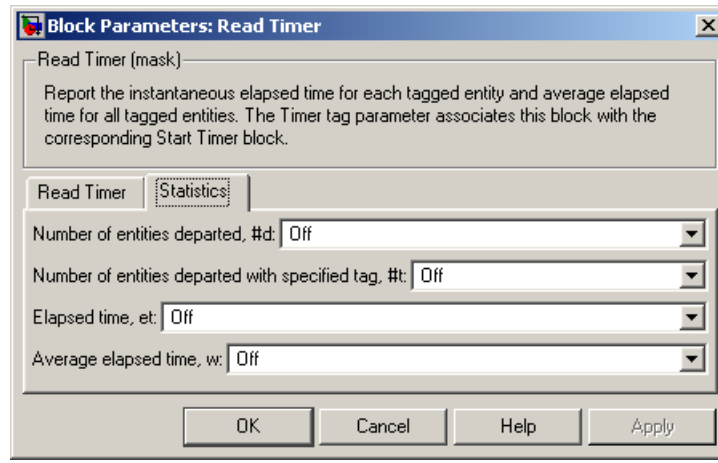
If entity does not have tagged timer

Behavior of the block if an arriving entity does not possess a timer with the specified timer tag.

Statistics Tab

These parameters determine whether 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
: Controls the presence and behavior of the signal output port labeled **#d**.

Number of entities departed with specified tag
: Controls the presence and behavior of 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.

Report elapsed time
: Controls the presence of the signal output port labeled **et**.

Report average elapsed time
: Controls the presence and behavior of the signal output port labeled **w**.

Examples

- "Basic Procedure for Using Timer Blocks" in the SimEvents user guide documentation

- "Timing Multiple Entity Paths with One Timer" in the SimEvents user guide documentation
- "Restarting a Timer from Zero" in the SimEvents user guide documentation
- "Timing Multiple Processes Independently" in the SimEvents user guide documentation

See Also

Start Timer

"Using Timers" in the SimEvents user guide documentation

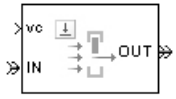
Purpose

Permit 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 block's entity input port is unavailable. By definition, the gate's opening permits one pending entity to arrive if able to advance immediately to the next block.

No simulation time passes between the opening and subsequent closing of the gate; that is, 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. Possible types are

- Edges in a trigger signal
- Changes in the numerical value of a signal
- Function calls

For more details, see "Opening a Gate Instantaneously" in the SimEvents user guide documentation.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Signal Input Ports

Label	Description
tr	Trigger signal that indicates when to open the gate. This port appears only if you set **Open gate upon** to `Trigger from port tr`.
vc	Control signal whose numerical changes indicate when to open the gate. This port appears only if you set **Open gate upon** to `Change in signal from port vc`.
fcn	Function-call signal that indicates when to open the gate. This port appears only if you set **Open gate upon** to `Function call from port fcn`.

Entity Output Ports

Label	Description
OUT	Port for departing entities.

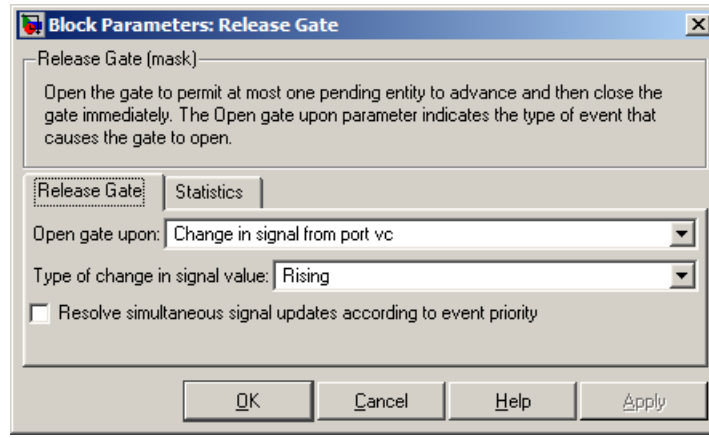
Signal Output Ports

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, which is in effect from the start of the simulation until the first update by the block, is 0.

Dialog Box

Release Gate Tab



Open gate upon
: Determines the type of event that causes the gate to open instantaneously.

Trigger type
: Determines whether rising, falling, or either type of trigger edge causes the gate to open. This field appears 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. This field appears only if you set **Open gate upon** to `Change in signal from port vc`.

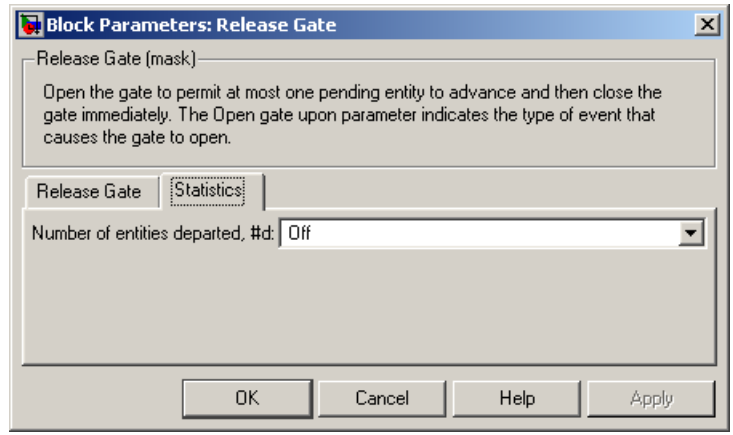
Resolve simultaneous signal updates according to event priority
: Select this option to explicitly prioritize the gate-opening event, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority `SYS1` on the event calendar. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation.

Event priority

The priority of the gate-opening event, relative to other simultaneous events in the simulation. For details, see "Specifying Event Priorities to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears only if you select **Resolve simultaneous signal updates according to event priority**.

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

Controls the presence and behavior of the signal output port labeled **#d**.

Examples

- "Example: Synchronizing Service Start Times with the Clock" in the SimEvents user guide documentation
- "Example: First Entity as a Special Case" in the SimEvents user guide documentation

See Also

Enabled Gate

"Regulating Arrivals Using Gates" in the SimEvents user guide documentation

Replicate

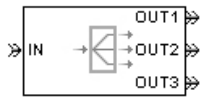
Purpose

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. For details about timeout events, see "Forcing Departures Using Timeouts" in the SimEvents user guide documentation.

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.

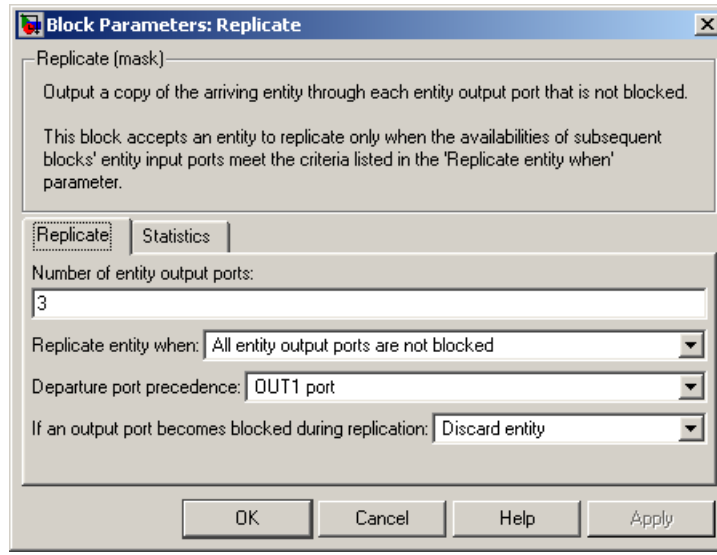
Signal Output Ports

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.

Dialog Box

Replicate Tab



Number of entity output ports

Determines how many entity output ports the block has; that is, the maximum number of copies the block makes for each arriving entity. This parameter must be a literal value, not a variable or an expression requiring evaluation.

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

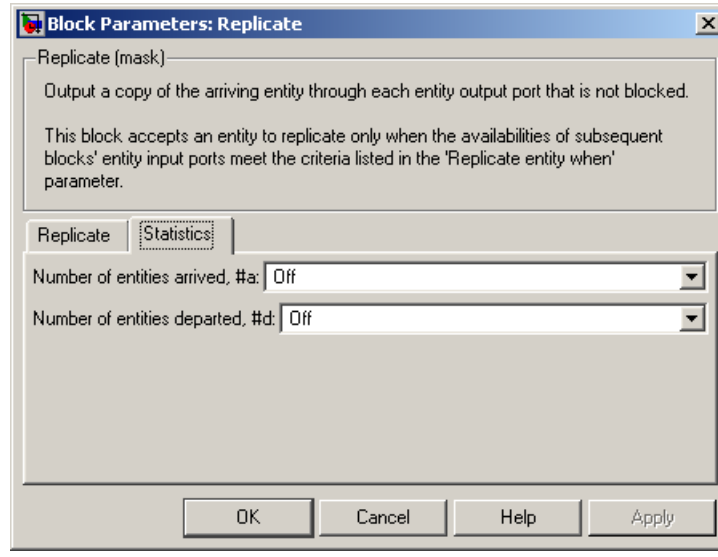
A nonnegative integer that initializes the random number generator used to determine the output sequence. This field

appears 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. This field appears only if you set **Replicate entity when** to `All entity output ports are not blocked`.

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 arrived
Controls the presence and behavior of the signal output port labeled **#a**.

Number of entities departed
Controls the presence and behavior of the signal output port labeled **#d**.

Examples

- "Example: Waiting Time in LIFO Queue" in the SimEvents user guide documentation
- Variable Entity Replication demo
- Ethernet Local Area Network demo

See Also

Event-Based Entity Generator, Path Combiner

"Replicating Entities on Multiple Paths" in the SimEvents user guide documentation

Purpose

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 common problems and troubleshooting tips, see "Unexpected Use of Old Value of Signal" in the SimEvents user guide documentation.

  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. For details, see "Assigning Event Priorities" and "Processing Sequence for Simultaneous Events" in the SimEvents user guide documentation.

Occurrence of Timeout Event

If the timeout event occurs for a specific entity, then that entity attempts to depart from a **TO** entity output port of the storage block in which it resides. To configure a block so that it has a **TO** port, select the **Enable TO port for timed-out entities** parameter in the block's dialog box. If the timeout event occurs while the entity is in a block that has no **TO** port, then the Schedule Timeout block's **If entity has no destination when timeout occurs** parameter indicates whether the simulation halts with an error message, or discards the entity while issuing a warning.

To cancel a timeout event before it occurs, use the Cancel Timeout 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 port appears 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.

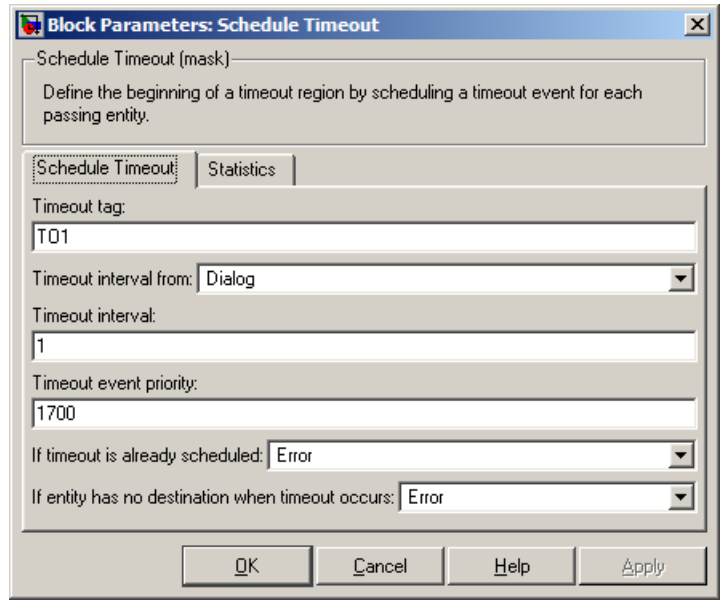
Signal Output Ports

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, which is in effect from the start of the simulation until the first update by the block, is 0.

Dialog Box

Schedule Timeout Tab



Timeout tag
Name of the timeout to associate with each entity.

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. This field appears 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. This field appears only if you set **Timeout interval from** to `Attribute`.

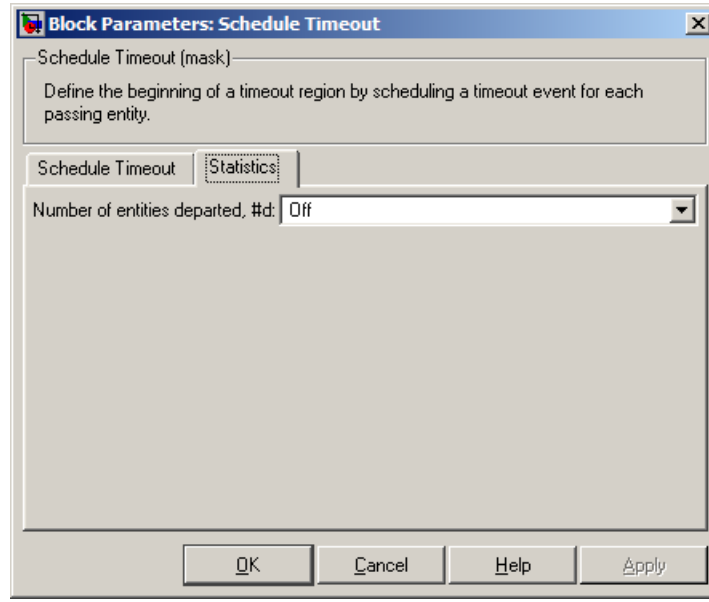
Timeout event priority
: The priority of the timeout event, relative to other simultaneous events in the simulation.

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 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
Controls the presence and behavior of the signal output port labeled **#d**.

Examples

- "Basic Example Using Timeouts" in the SimEvents user guide documentation
- "Defining Entity Paths on Which Timeouts Apply" in the SimEvents user guide documentation
- "Example: Dropped and Timed-Out Packets" in the SimEvents user guide documentation
- "Example: Rerouting Timed-Out Entities to Expedite Handling" in the SimEvents user guide documentation
- "Example: Limiting the Time Until Service Completion" in the SimEvents user guide documentation

See Also

Cancel Timeout

"Forcing Departures Using Timeouts" in the SimEvents user guide documentation

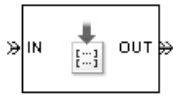
Purpose

Assign data to entity

Library

Attributes

Description



This block accepts an entity, assigns data to it, and then outputs it. Assigned data is stored in attributes of the entity, where each attribute has a name and a value. See “Attribute Value Support” in the SimEvents user guide documentation to learn what kind of data an attribute can store.

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.

Manipulating the Rows of the Table

Each attribute corresponds to a row in the table on the **Set Attribute** tab in the block’s dialog box. Buttons to the left of the table let you manipulate rows in the table:

- To add a new row to the table, click the Add button . By default, the attribute name in the new row is unique within the table.
- To duplicate a row, select it and click the Copy button .
- To remove a row, select it and click the Delete button .

Note The dialog box does not ask you to confirm the deletion and does not offer an undo operation. However, if you delete a row by mistake, you can click **Cancel** to ignore unapplied changes.

Note Deleting a row and applying the change might affect signal input ports corresponding to other rows of the table. For example, if the block has a signal input port **A2** and you delete the row marked **A1**, then the block renames **A2** as **A1**. Check that any signal that connects to the renamed port is still connected as you expect.

Configuring Individual Rows to Set Particular Attributes

Within each row, you can specify properties of the attribute that corresponds to that row.

Assigning a Constant Value Using the Block Parameters Dialog Box

1. Select the **Name** field and type the name of the attribute you want to set. All names that would be valid variable names in the MATLAB language, except `nan` and `inf` (with any use of case), are valid as attribute names. To determine whether a name is a valid variable name, use the `isvarname` function.
2. Set **Value From** to `Dialog`.
3. Select the **Value** field and enter the value for the attribute.
4. Determine if the constant value you typed is a vector that you want the block to interpret a one-dimensional array rather than a multidimensional array. If yes, select **Treat Vector as 1-D**.

Assigning a Value Using an Input Signal

1. Select the **Name** field and type the name of attribute you want to set. All names that would be valid variable names in the MATLAB language, except `nan` and `inf` (with any use of case), are valid as attribute names. To determine whether a name is a valid variable name, use the `isvarname` function.

2 Set **Value From** to `Signal port`. The **Value** and **Treat Vector as 1-D** fields become disabled.

3 Click **OK** or **Apply**. The block now has a signal input port whose label matches the label to the left of the row of the table (**A1**, **A2**, and so on).

4 Connect a signal to the **A*x*** input port. During the simulation, the block assigns the value of this signal to the attribute. The signal must be sample-based instead of frame-based.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Signal Input Ports

Label	Description
A*x*	Data to assign for the attribute specified in the **A*x*** row of the table in the dialog box. This port appears only if you set **Value From** in the **A*x*** row to `Signal port`.

Entity Output Ports

Label	Description
OUT	Port for departing entities, with data assigned to them.

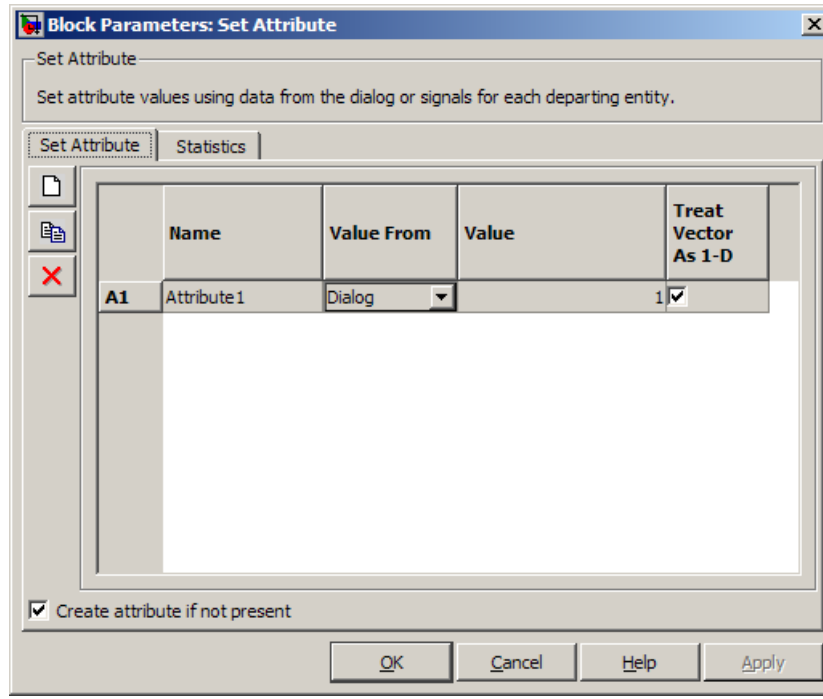
Signal Output Ports

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, which is in effect from the start of the simulation until the first update by the block, is 0.

Dialog Box

Set Attribute Tab



Name
The name of the attribute to set.

Value From
Determines whether the data for attribute values comes from the dialog or a signal.

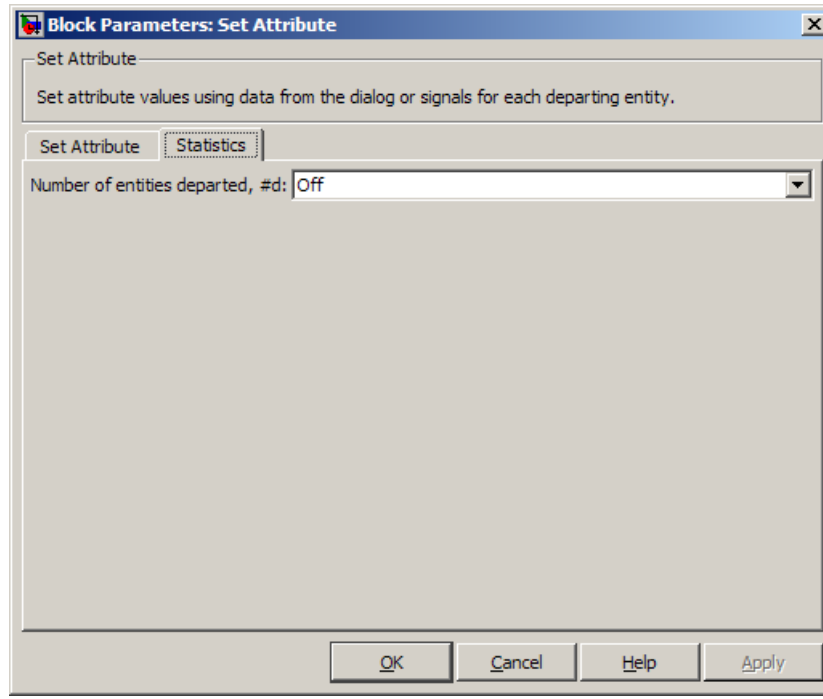
Value
The value to assign to the attribute. This field is active only if you set **Value From** to `Dialog`.

Treat Vector as 1-D
This option affects attributes whose **Value From** parameter is set to `Dialog` and whose **Value** parameter evaluates to an N-element row or column vector. Selecting this option causes the block to assign the attribute as a vector of length N. Otherwise, the block assigns the attribute as a multidimensional array.

Create attribute if not present
Selecting this option enables the block to define new attributes. Otherwise, the block issues an error if any attribute named in the table does not already exist.

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
: Controls the presence and behavior of the signal output port labeled **#d**.

Examples

- "Example: Setting Attributes" in the SimEvents user guide documentation
- "Example: Using an Attribute to Select an Output Port" in the SimEvents getting started documentation
- "Example: Round-Robin Approach to Choosing Inputs" in the SimEvents getting started documentation
- "Adding Event-Based Behavior" in the SimEvents getting started documentation

- “Example: Rerouting Timed-Out Entities to Expedite Handling” in the SimEvents user guide documentation

See Also

Get Attribute

“Setting Attributes of Entities” in the SimEvents user guide documentation

Set Attribute (Obsolete)

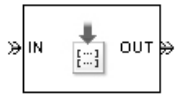
Purpose

Assign data to entity

Library

Attributes

Description



Note The Set Attribute block in the `simeventsattributes1` library will be removed in a future release. Use the Set Attribute block in the `simeventsattributes2` library instead.

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.

You can assign up to four attributes with a single instance of this block. For each attribute you want to set, configure one of the tabs named **A1**, **A2**, **A3**, and **A4** in the block's dialog box using one of these procedures:

Assigning a Constant Value Using This Dialog Box

1. Set **Attribute assignment** to `Specify via dialog`.
2. Specify the name of attribute you want to set, in the **Attribute name** field. All names that would be valid variable names in the MATLAB language, except `nan` and `inf` (with any use of case), are valid as attribute names. To determine whether a name is a valid variable name, use the `isvarname` function.
3. Specify the constant value for the attribute in the **Value** field.

Assigning a Value Using an Input Signal

1. Set **Attribute assignment** to `From signal port A`*x*, where *x* is 1, 2, 3, or 4.
2. Specify the name of attribute you want to set, in the **Attribute name** field. All names that would be valid variable names in the MATLAB language, except `nan` and `inf` (with any use of case), are valid as

attribute names. To determine whether a name is a valid variable name, use the `isvarname` function.

3. Click **OK** or **Apply**. The block now has a signal input port labeled **A*x***.

4. Connect a signal to the **A*x*** input port. During the simulation, the block assigns the value of this signal to the attribute.

If you want to set fewer than the maximum number of attributes, then you can deactivate tabs that you are not using by setting those tabs' **Attribute assignment** parameters to `Off`.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Signal Input Ports

Label	Description
A*x*	Data to assign for the attribute specified on the **A*x*** tab of the dialog. This port appears only if you set **Attribute assignment** on the **A*x*** tab to `From signal port A`*x*.

Entity Output Ports

Label	Description
OUT	Port for departing entities, with data assigned to them.

Signal Output Ports

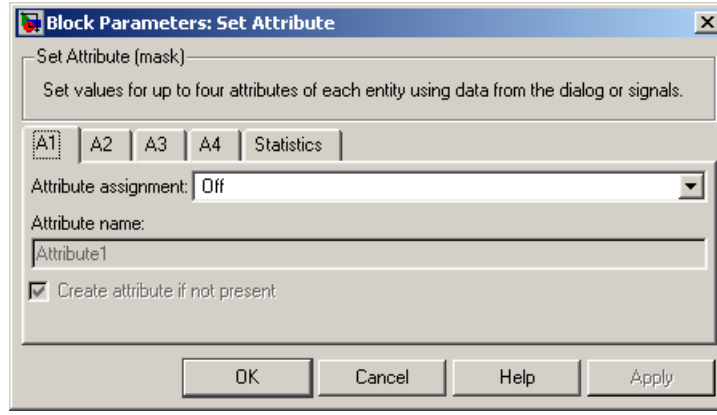
Label	Description	Time of Update When Statistic Is On
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure

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

Dialog Box

A1, A2, A3, A4 Tabs

The **A1**, **A2**, **A3**, and **A4** tabs have similar parameter choices. By assigning different values to the parameters, you can configure this block to assign up to four different attributes for each entity that the block processes.



Attribute assignment

Determines whether the data for attribute values comes from the dialog or a signal. Choosing `Off` indicates that you are not using this tab of the dialog and makes the parameters below inactive or invisible.

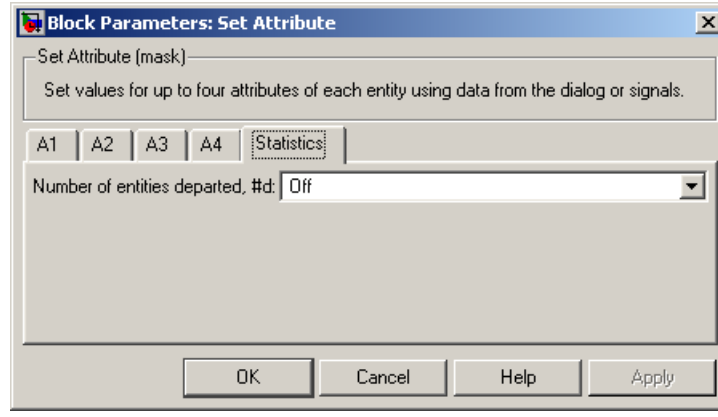
Attribute name
The name of the attribute to set.

Value
The value to assign to the attribute named above. This parameter appears only if you set **Attribute assignment** to `Specify via dialog`.

Create attribute if not present
Selecting this option enables the block to define new attributes. Otherwise, the block issues an error if the attribute named above does not already exist.

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
Controls the presence and behavior of the signal output port labeled **#d**.

Set Attribute (Obsolete)

Examples

- "Example: Setting Attributes" in the SimEvents user guide documentation
- "Example: Using an Attribute to Select an Output Port" in the SimEvents getting started documentation
- "Example: Round-Robin Approach to Choosing Inputs" in the SimEvents getting started documentation
- "Adding Event-Based Behavior" in the SimEvents getting started documentation
- "Example: Rerouting Timed-Out Entities to Expedite Handling" in the SimEvents user guide documentation

See Also

Set Attribute, Get Attribute

"Setting Attributes of Entities" in the SimEvents user guide documentation

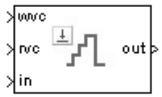
Purpose

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. For an example

that uses this block in a feedback loop, see the Dynamic Voltage Scaling Using Online Gradient Estimation demo.

Ports

Signal Input Ports

Label	Description
wts	Signal whose updates cause write events. This port appears only if you set **Write to memory upon** to `Sample time hit from port wts`.
wtr	Trigger signal whose edges cause write events. This port appears only if you set **Write to memory upon** to `Trigger from port wtr`.
wvc	Signal whose numerical changes in value cause write events. This port appears only if you set **Write to memory upon** to `Change in signal from port wvc`.
wfcn	Function-call signal that causes write events. This port appears only if you set **Write to memory upon** to `Function call from port wfcn`.
rts	Signal whose updates cause read events. This port appears only if you set **Read from memory upon** to `Sample time hit from port rts`.
rtr	Trigger signal whose edges cause read events. This port appears only if you set **Read from memory upon** to `Trigger from port rtr`.
rvc	Signal whose numerical changes in value cause read events. This port appears only if you set **Read from memory upon** to `Change in signal from port rvc`.
rfcn	Function-call signal that causes read events. This port appears only if you set **Read from memory upon** to `Function call from port rfcn`.
in	Signal to be resampled and/or delayed.

Signal Output Ports

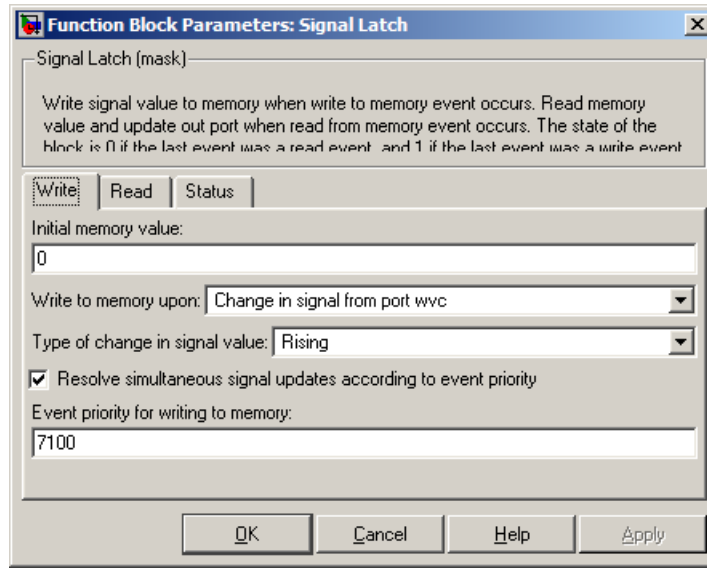
Label	Description	Time of Update When Statistic Is On	Order of Update	Initial Value
st	0 or 1, depending on whether the block more recently processed a read or write event.	Upon write events and upon read events	1	0
mem	The value of the block's internal memory when a write event occurs.	Upon write events	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.

Dialog Box

Write Tab



Initial memory value
The value in the block's internal memory before the first write event occurs.

Write to memory upon
The type of signal-based event or function call that causes a write event.

Trigger type
Determines whether rising, falling, or either type of trigger edge causes a write event. This field appears only if you set **Write to memory upon** to `Trigger from port wtr`.

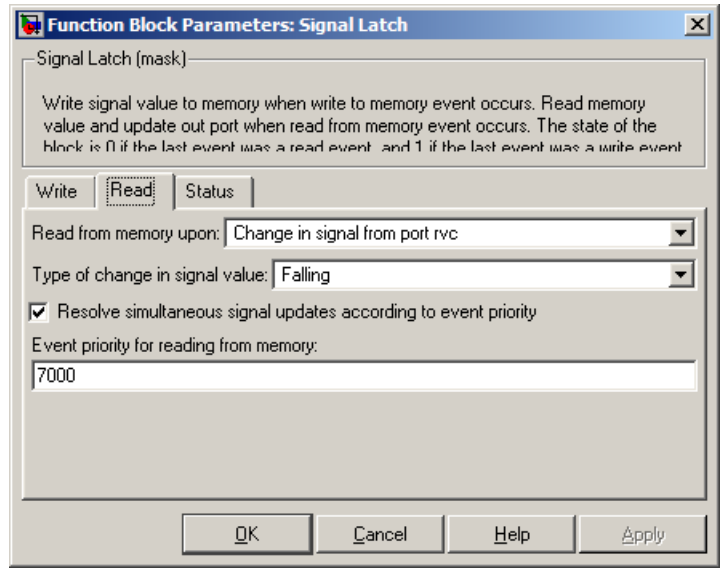
Type of change in signal value
Determines whether rising, falling, or either type of value change causes a write event. This field appears only if you set **Write to memory upon** to `Change in signal from port wvc`.

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. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation.

Note If this block has a function-call input, you might need to select this option to prevent latency in the signal input.

Event priority for writing to memory
The priority of the write event, relative to other simultaneous events in the simulation. For details, see "Specifying Event Priorities to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears only if you select **Resolve simultaneous signal updates according to event priority** on this tab.

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
Determines whether rising, falling, or either type of trigger edge causes a read event. This field appears 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. This field appears 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

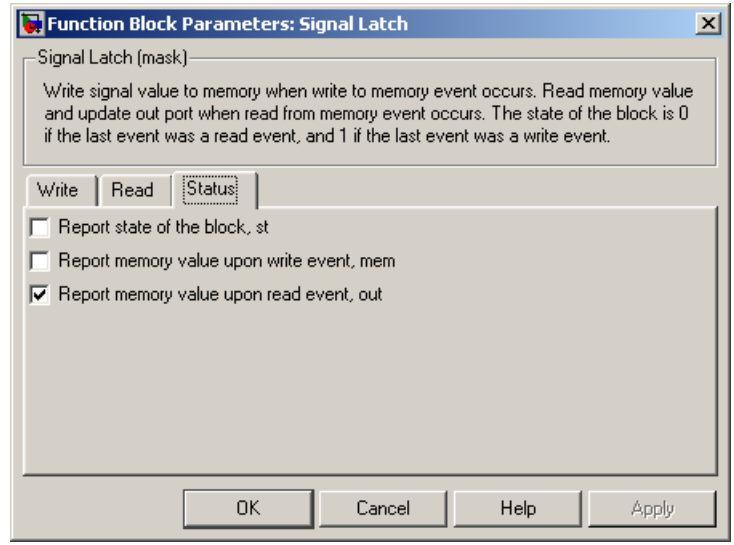
immediately upon detecting the signal-based event that causes it. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears only if you set **Read from memory upon** to an option other than `Write to memory event`.

Note If this block has a function-call input, you might need to select this option to prevent latency in the signal input.

Event priority for reading from memory

The priority of the read event, relative to other simultaneous events in the simulation. For details, see "Specifying Event Priorities to Resolve Simultaneous Signal Updates". This field appears only if you select **Resolve simultaneous signal updates according to event priority** on this tab.

Status Tab



Report state of the block
Controls the presence of the signal output port labeled **st**.

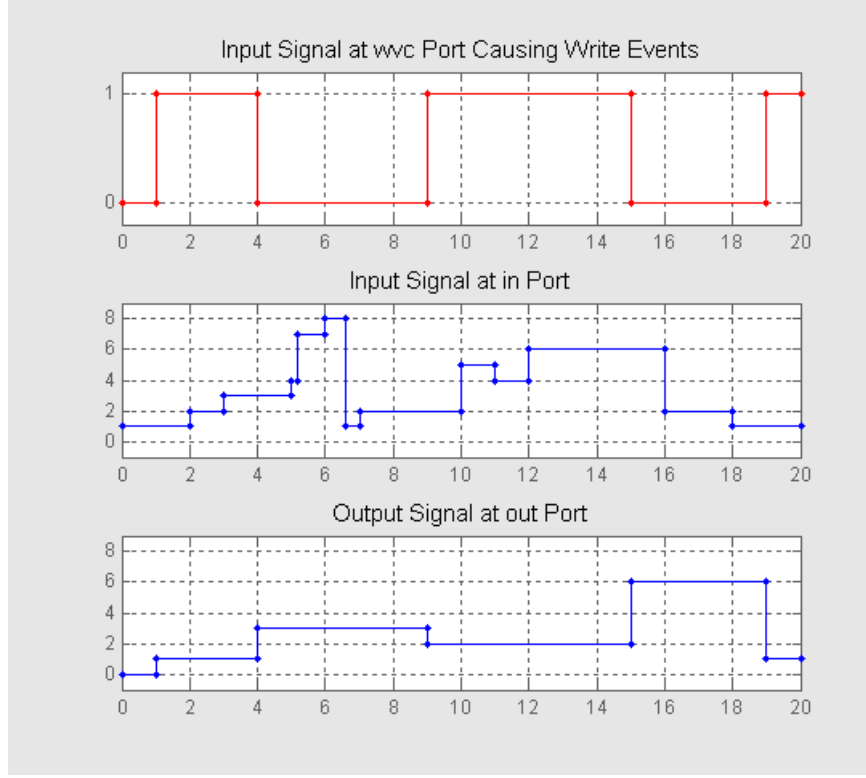
Report memory value upon write event
Controls the presence of the signal output port labeled **mem**.

Report memory value upon read event
Controls the presence of the signal output port labeled **out**.

Examples

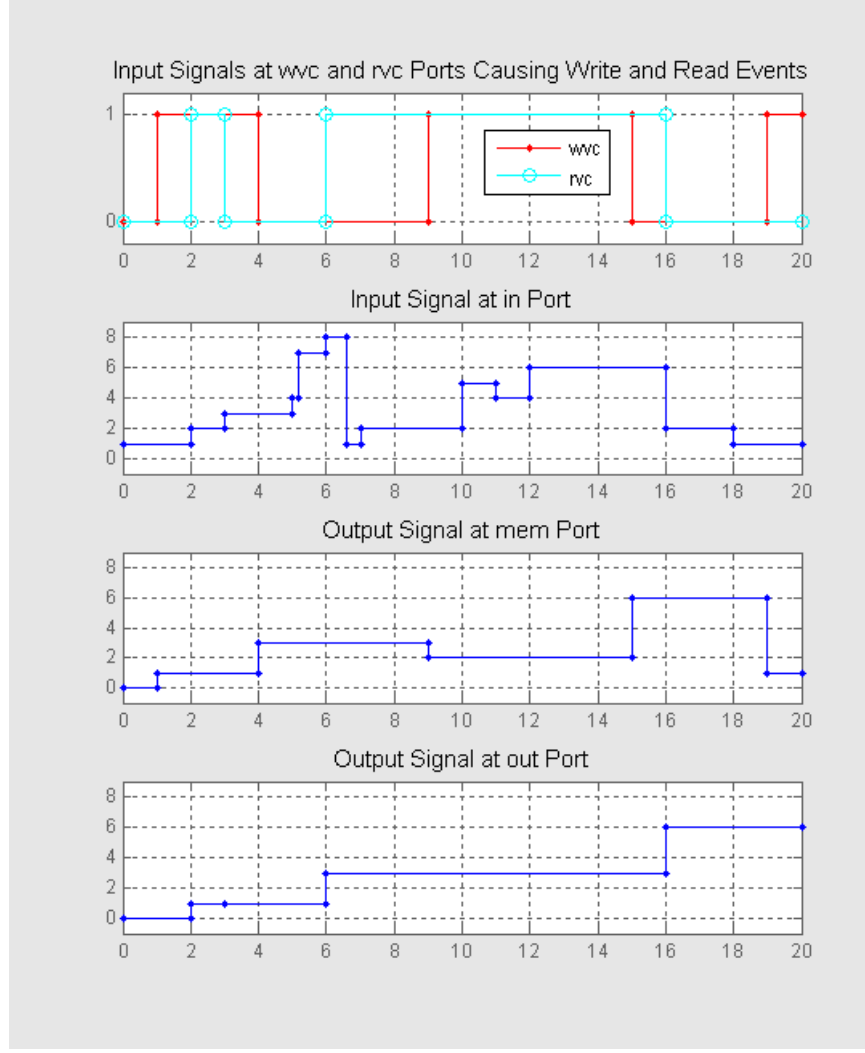
Reading from Memory Upon Each Write Event

In the plot below, the output signal reflects values of the input signal upon each rising or falling value of the **wvc** signal. Between successive write events, the output signal maintains the value from the most recent write event. Before the first write event, the output signal is 0 because of the initial memory value.



Independent Read and Write Events

In the plot below, the **mem** signal reflects values of the input signal upon each rising or falling value of the **wvc** signal, while the **out** signal reflects values of the **mem** signal upon each rising or falling value of the **rvc** signal.


Input Signals at wvc and rvc Ports Causing Write and Read Events
wvc
rvc
Input Signal at in Port
Output Signal at mem Port
Output Signal at out Port

For examples showing the use of this block in a model, see

- “Example: Creating a Random Signal for Switching” in the SimEvents user guide documentation
- “Generating Random Time-Based Signals” in the SimEvents user guide documentation
- “Example: Resampling a Signal Based on Events” in the SimEvents user guide documentation
- “Example: Detecting Collisions by Comparing Events” in the SimEvents user guide documentation
- “Example: Compound Switching Logic” in the SimEvents user guide documentation

See Also

Data Store Memory, Data Store Read, Data Store Write

Signal Scope

Purpose

Plot data from signal

Library

SimEvents Sinks

Description

This block creates a plot using data from a signal. The plot is particularly appropriate for data arising from discrete-event simulations or data related to entities because the plot can include zero-duration values.

The data for the vertical axis comes from the signal connected to the block's signal input port labeled **in**.

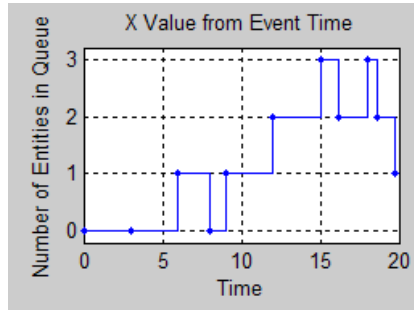
The **Plot type** parameter on the **Plotting** tab determines whether and how the block connects the points that it plots. For details, see "Connections Among Points in Plots" in the SimEvents user guide documentation.

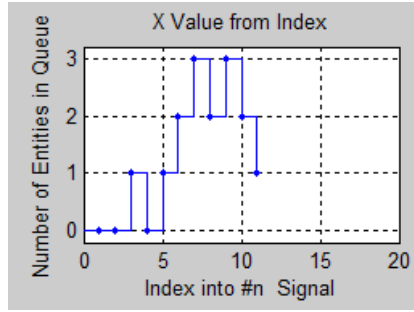
Selecting Data for the Horizontal Axis

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.

Signal Output Ports

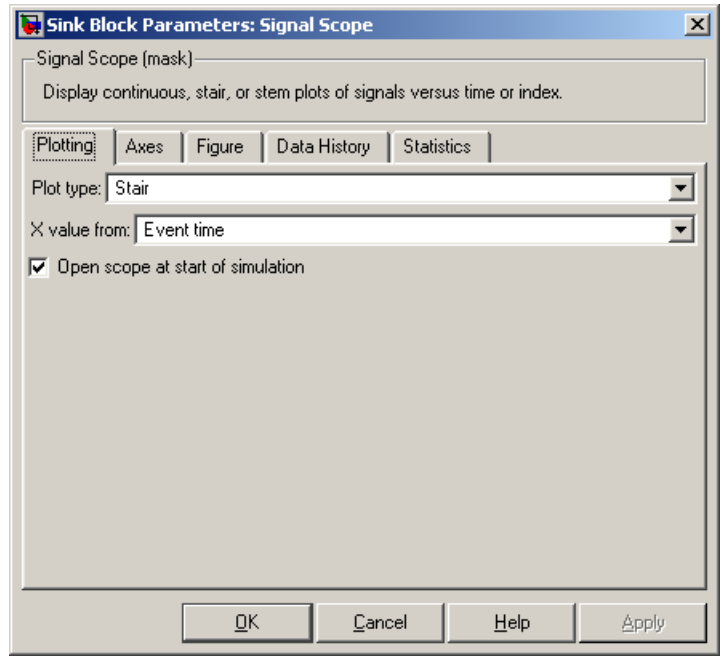
Label	Description
#c	Number of points the block has plotted.

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

Dialog Box

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

Plotting Tab



Plot type

The presentation format for the data. See "Connections Among Points in Plots" in the SimEvents user guide documentation for details.

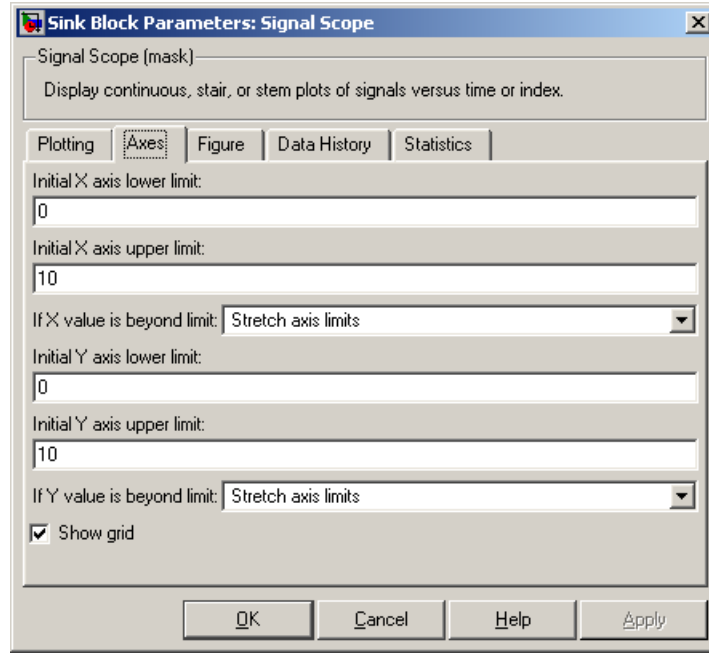
X value from

Source of data for the plot's horizontal axis. See "Selecting Data for the Horizontal Axis" on page 4-218 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.

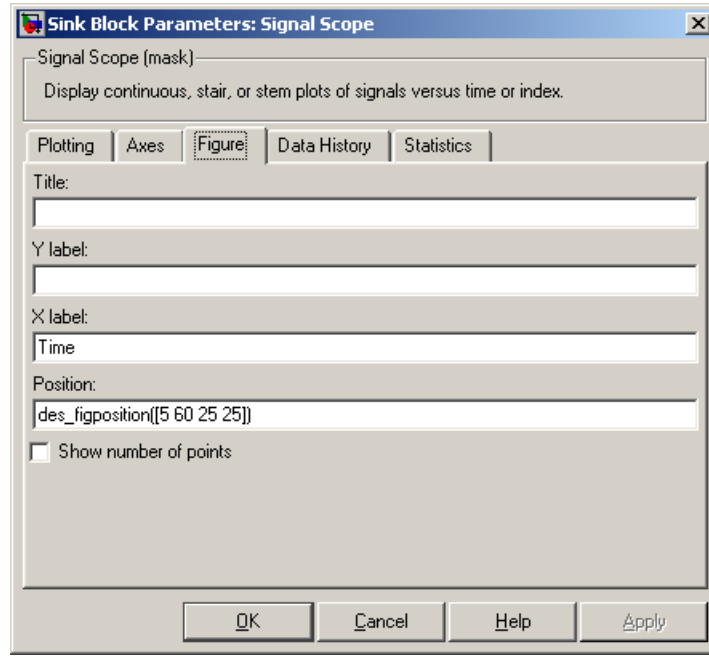
If X value is beyond limit
Determines how the plot changes if one or more X values are not within the limits shown on the X axis. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

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. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

Show grid
Toggles the grid on and off.

Figure Tab



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

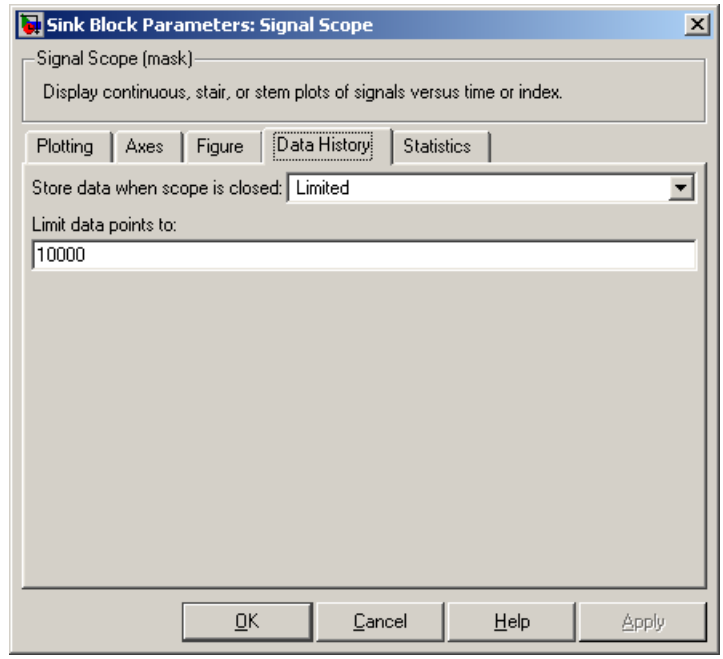
Y label
Text that appears to the left of the vertical axis.

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

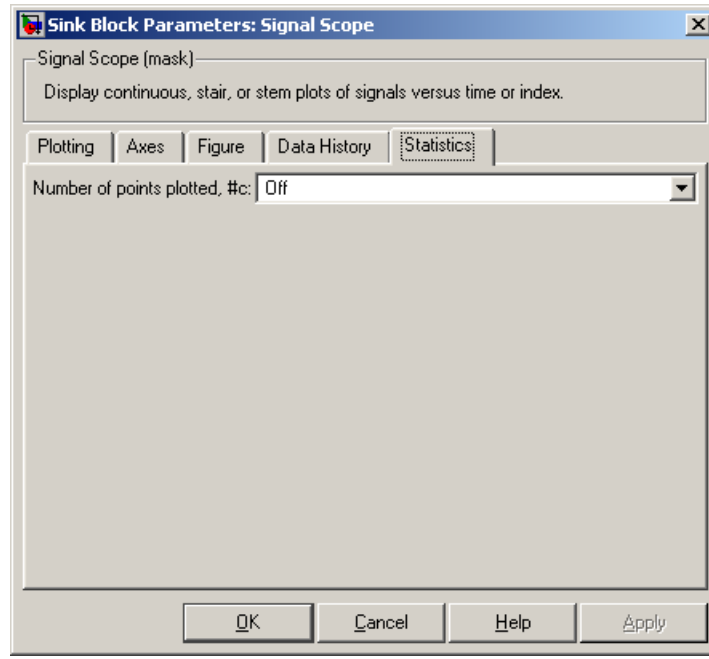


Store data when scope is closed
Select `Unlimited` to have the block cache all data for future viewing, `Limited` to cache a portion of the most recent data, and `Disabled` to avoid caching undisplayed data.

Limit data points to
The number of data points the block caches, using the most recent data. This field appears only if you set **Store data when scope is closed** to `Limited`.

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 points plotted

Controls the presence and behavior of the signal output port labeled **#c**.

Examples

- "Building a Simple Discrete-Event Model" and "Observations from Plots" in the SimEvents getting started documentation
- "Example: Using Servers in Shifts" in the SimEvents user guide documentation

- “Example: Choosing the Shortest Queue” in the SimEvents user guide documentation

See Also

X-Y Signal Scope, Attribute Scope

“Plotting Data” in the SimEvents user guide documentation

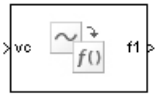
Purpose

Convert signal-based events into function calls

Library

Event Translation

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 port appears only if you select **Resolve simultaneous signal updates according to event priority**, and then set **Function-call delay from** to `Signal port t`.
ts	Primary criterion is satisfied when the application updates the signal at this port. This port appears only if you set **Generate function call only upon** to `Sample time hit from port ts`.
tr	Primary criterion is satisfied when the signal has a rising or falling edge, depending on the **Trigger type** parameter. This port appears only if you set **Generate function call only upon** to `Trigger from port tr`.
vc	Primary criterion is satisfied when the signal at this port increases or decreases, depending on the **Type of change in signal value** parameter. This port appears only if you set **Generate function call only upon** to `Change in signal from port vc`.
fcn	Primary criterion is satisfied when this input port detects a function call. This port appears 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 input port appears 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 input port appears only if you select **Suppress function call f2 if enable signal e2 is not positive**.

Signal Output Ports

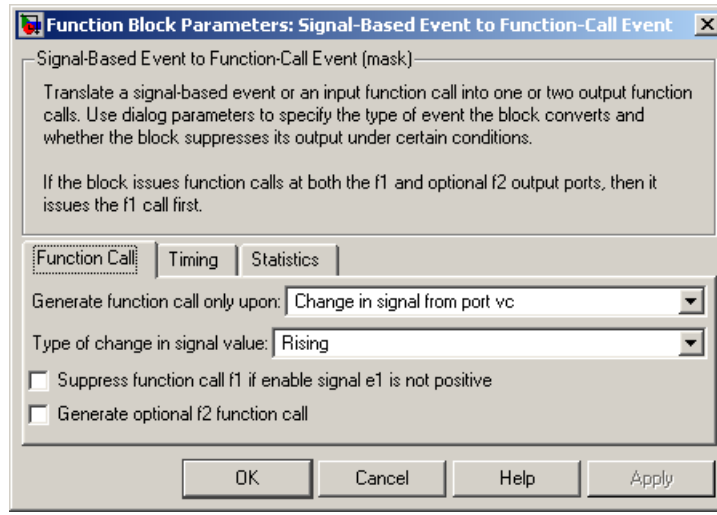
Label	Description	Order of Update
f1	Function call, possibly contingent on **e1** input signal	1
f2	Function call, possibly contingent on **e2** input signal	2
#f1	Number of function calls the block has generated at the **f1** port during the simulation	3
#f2	Number of function calls the block has generated 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.

Dialog Box

Function Call Tab



Generate function call only upon

The primary criterion for determining when the block generates a function call. Optional secondary criteria are established by the **Suppress function call...** parameters below.

Trigger type

Determines whether rising, falling, or either type of trigger edge causes the block to generate a function call. This field appears only if you set **Generate function call only upon** to `Trigger from port tr`.

Type of change in signal value

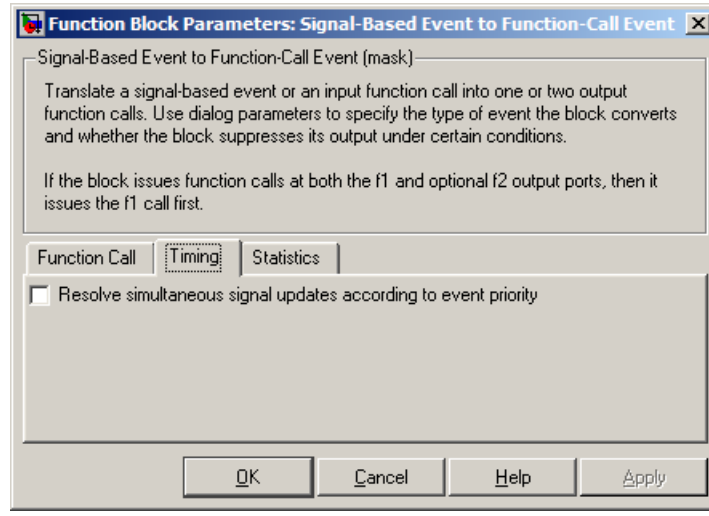
Determines whether rising, falling, or either type of value change causes the block to generate a function call. This field appears only if you set **Generate function call only upon** to `Change in signal from port vc`.

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. This field appears 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. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation.

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. For details, see "Specifying Event Priorities to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears only if you select **Resolve simultaneous signal updates according to event priority**.

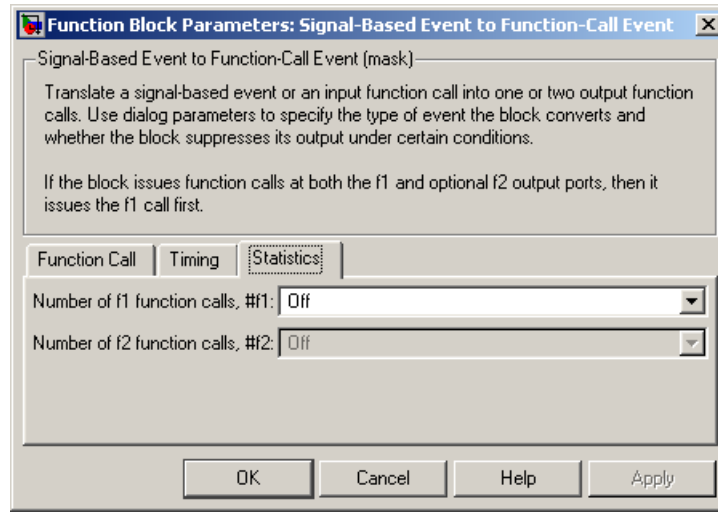
Function-call delay from

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. This field appears 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. This field appears 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

Controls the presence and behavior of the signal output port labeled **#f1**.

Number of f2 function calls

Controls the presence and behavior of 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.

Examples

- "Example: Detecting Changes in the Last-Updated Signal" in the SimEvents user guide documentation
- "Example: Using a #n Signal as a Trigger" in the SimEvents user guide documentation

See Also

Entity Departure Event to Function-Call Event

"Manipulating Events" in the SimEvents user guide documentation

Signal-Based Function-Call Event Generator

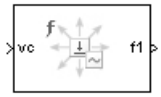
Purpose

Generate function-call events in response to signal-based events

Library

Generators / Event Generators

Description



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 Event to Function-Call Event block, which offers more flexibility.

Ports

Signal Input Ports

Label	Description
ts	The block generates a function call when the application updates the signal at this port. This port appears only if you set **Generate function call only upon** to `Sample time hit from port ts`.
tr	The block generates a function call when the signal has a rising or falling edge, depending on the **Trigger type** parameter. This port appears only if you set **Generate function call only upon** to `Trigger from port tr`.
vc	The block generates a function call when the signal at this port increases or decreases, depending on the **Type of change in signal value** parameter. This port appears only if you set **Generate function call only upon** to `Change in signal from port vc`.
fcn	The block generates a function call when this input port detects a function call. This port appears 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.

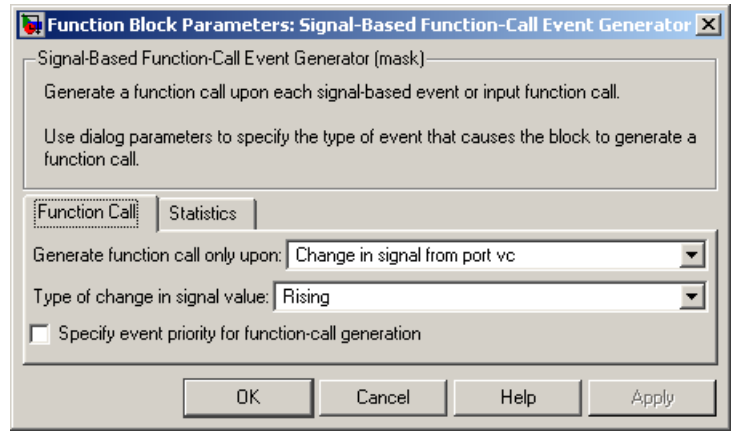
Signal Output Ports

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.

Dialog Box

Function Call Tab



Generate function call only upon

The primary criterion for determining when the block generates a function call. Optional secondary criteria are established by the **Suppress function call...** parameters below.

Trigger type

Determines whether rising, falling, or either type of trigger edge causes the block to generate a function call. This field appears

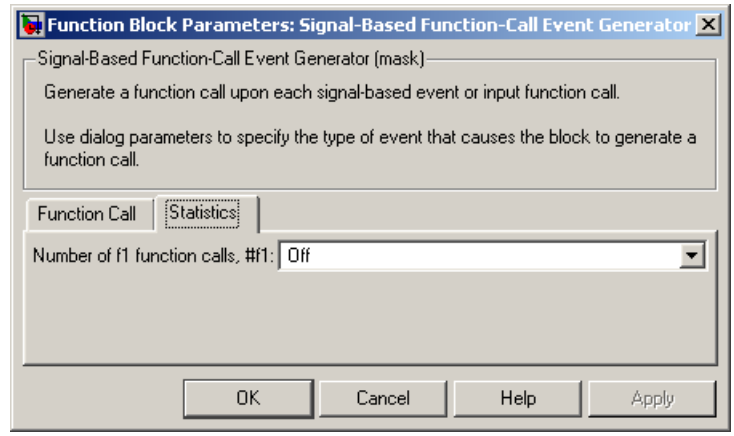
only if you set **Generate function call only upon** to `Trigger from port tr`.

Type of change in signal value
Determines whether rising, falling, or either type of value change causes the block to generate a function call. This field appears 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. For details, see "Choosing How to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation.

Event priority
The priority of the function-call event, relative to other simultaneous events in the simulation. For details, see "Specifying Event Priorities to Resolve Simultaneous Signal Updates" in the SimEvents user guide documentation. This field appears only if you select **Resolve simultaneous signal updates according to event priority**.

Statistics Tab



Number of f1 function calls

Controls the presence and behavior of the signal output port labeled **#f1**.

Examples

- "Example: Calling a Stateflow Block Upon Changes in Server Contents" in the SimEvents user guide documentation
- "Example: Counting Events from Multiple Sources" in the SimEvents user guide documentation

See Also

Signal-Based Event to Function-Call Event

"Generating Function-Call Events" in the SimEvents user guide documentation

Single Server

Purpose

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; see "Forcing Departures Using Timeouts" in the SimEvents user guide documentation for details about timeouts.

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. For common problems and troubleshooting tips, see "Unexpected Use of Old Value of Signal" in the SimEvents user guide documentation.

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. For details, see "Preempting an Entity in a Server" in the SimEvents user guide documentation.

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 port appears 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. This port appears 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 from the **OUT** port	3
#p	Number of entities that have been preempted from this block since the start of the simulation.	After entity departure from the **P** port	4
pe	A value of 1 indicates when the block tries and fails to output an entity, that is, an entity has completed its service, but cannot depart from the **OUT** port because the subsequent entity input port is unavailable.	After service completion and when entity departure is imminent	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 from the **OUT** or **TO** port	2

Signal Output Ports (Continued)

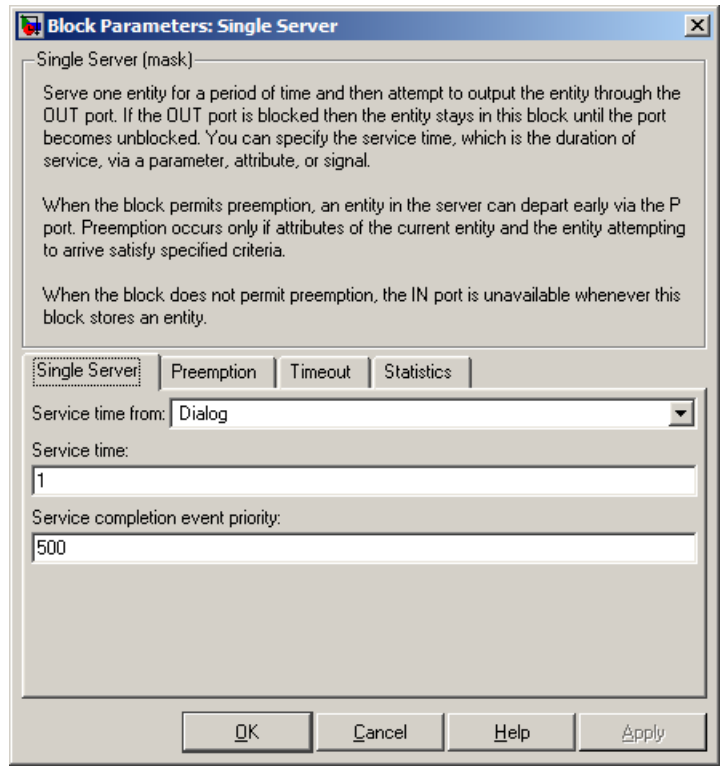
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
util	Utilization of the server, which is the fraction of simulation time spent storing an entity. At *T*=0, the utilization is 0 or 1 depending on whether the server contains an entity.	Performance considerations cause the block to suppress signal updates until specific occurrences cause updates. In `On` mode, updates occur after an entity departure from the **OUT** port and after an entity arrival. In `Upon stop or pause` mode, stopping or pausing the simulation causes an update.	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.

Dialog Box

Single Server Tab



Service time from

Determines whether the service time is computed from a parameter in this dialog box, an input signal, or an attribute of the entity being served.

Service time

The service time, in seconds, for all entities. This field appears only if you set **Service time from** to `Dialog`.

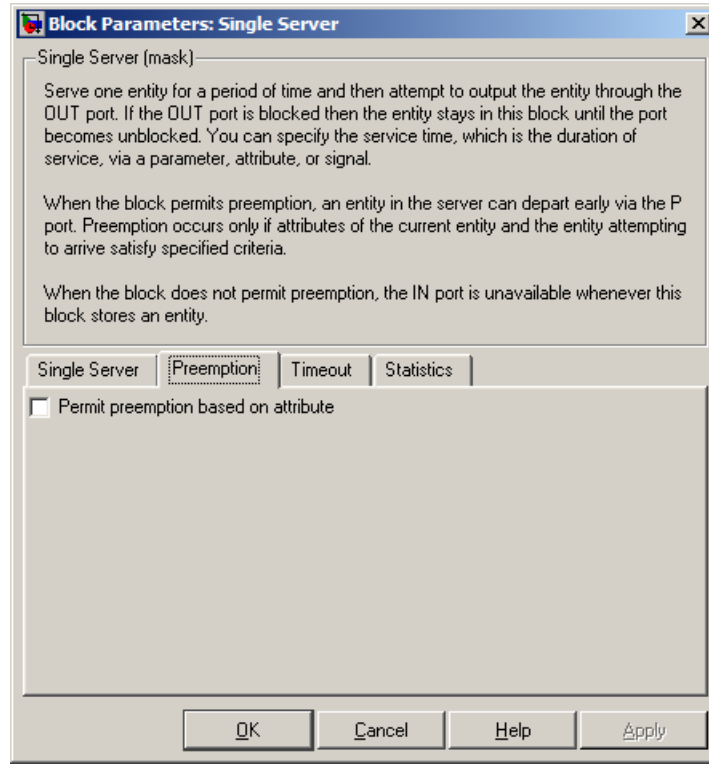
Attribute name

The name of the attribute whose value the block uses as the service time for an entity. This field appears 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 sets **Average wait** on the **Statistics** tab to `Off` 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. This field appears only if you select **Permit preemption based on attribute**.

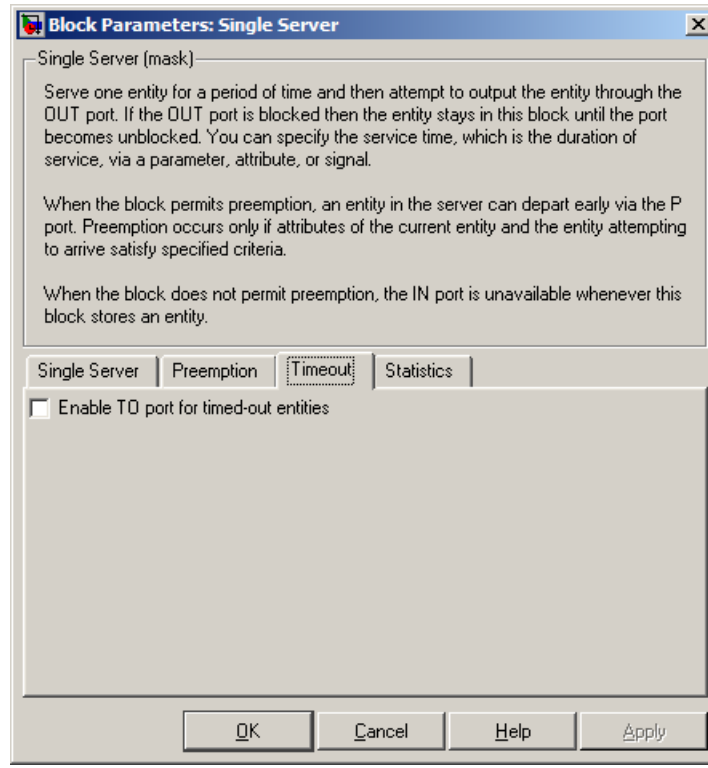
Sorting direction
Preemption occurs when the arriving entity has a strictly smaller (`Ascending`) or strictly larger (`Descending`) value of the attribute named above, compared to the attribute value of the entity in the server. This field appears only if you select **Permit preemption based on attribute**.

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. This field appears 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. This field appears 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. This field appears only if you select **Write residual service time to attribute**.

Timeout Tab

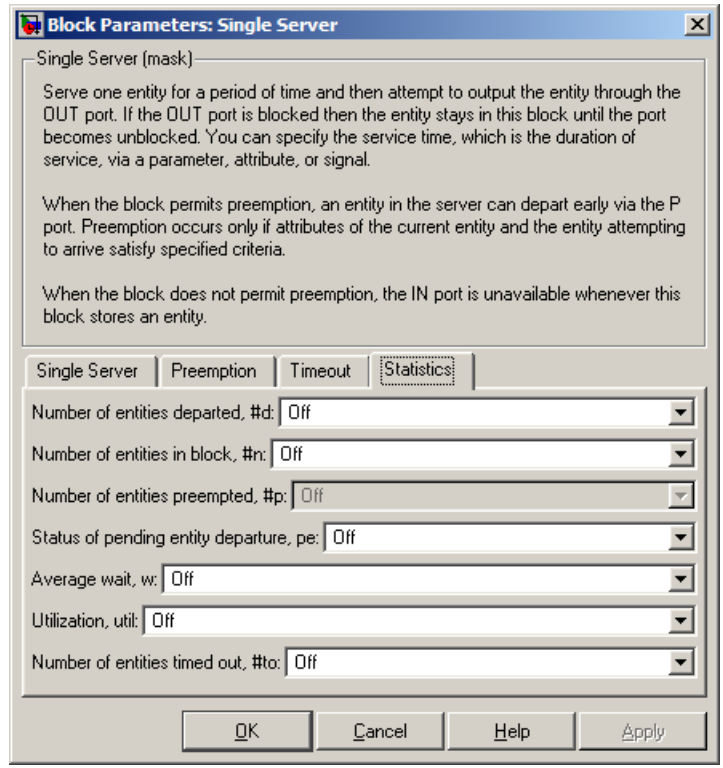


Enable TO port for timed-out entities

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 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

Controls the presence and behavior of the signal output port labeled **#d**.

Number of entities in block
Controls the presence and behavior of the signal output port labeled **#n**.

Number of entities preempted
Controls the presence and behavior of 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.

Status of pending entity departure
Controls the presence of the signal output port labeled **pe**.

Average wait
Controls the presence and behavior of 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
Controls the presence and behavior of the signal output port labeled **util**.

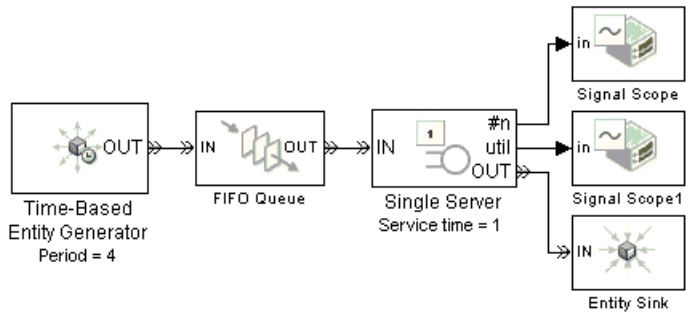
Number of entities timed out
Controls the presence and behavior of the signal output port labeled **#to**.

Examples

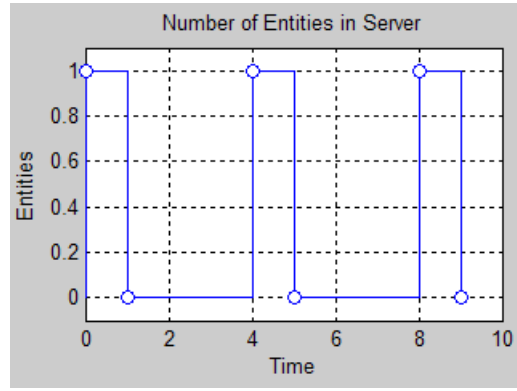
- "Building a Simple Discrete-Event Model" in the SimEvents getting started documentation
- "Example: Selecting the First Available Server" in the SimEvents getting started documentation
- "Constructs Involving Queues and Servers" in the SimEvents getting started documentation
- "Example: Using a Signal or an Attribute" in the SimEvents user guide documentation
- "Example: Using Servers in Shifts" in the SimEvents user guide documentation
- "Example: Preemption by High-Priority Entities" in the SimEvents user guide documentation

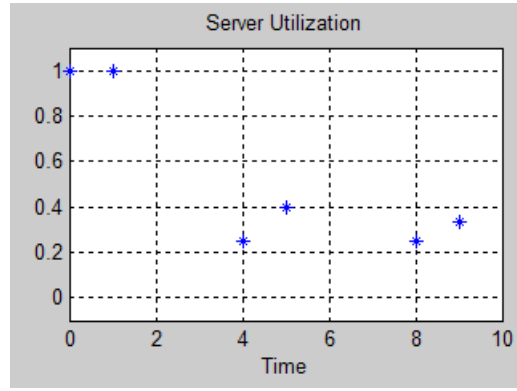
- "Example: Controlling Joint Availability of Two Servers" in the SimEvents user guide documentation
- "Example: Synchronizing Service Start Times with the Clock" in the SimEvents user guide documentation

The following example illustrates the timing of updates of the **util** signal, as described in Signal Output Ports on page 4-240.

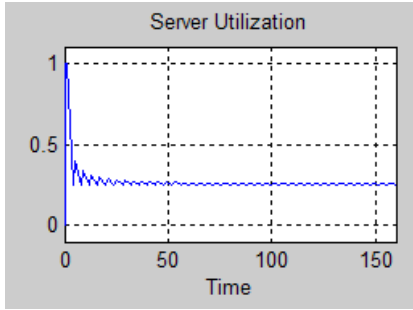


The server has idle periods that reduce its utilization. However, the server block recomputes the **util** signal only when the number of entities in the server changes. While the definition of utilization says that the utilization is less than 1 at time 3, the **util** signal remains at its previous 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, Infinite Server

"Basic Queues and Servers" in the SimEvents getting started documentation

Purpose

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 block. To read the value of the timer named in this block, reference the timer name in the Read Timer block. For more information about using this pair of blocks, see "Using Timers" in the SimEvents user guide documentation.

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.

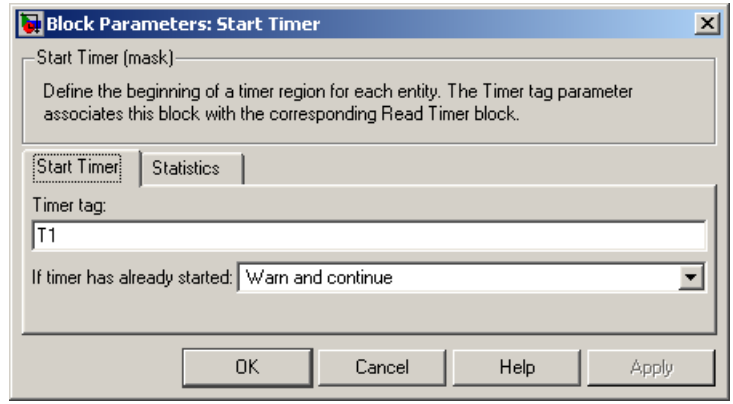
Signal Output Ports

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, which is in effect from the start of the simulation until the first update by the block, is 0.

Dialog Box

Start Timer Tab



Timer tag

Name of the timer to associate with each entity.

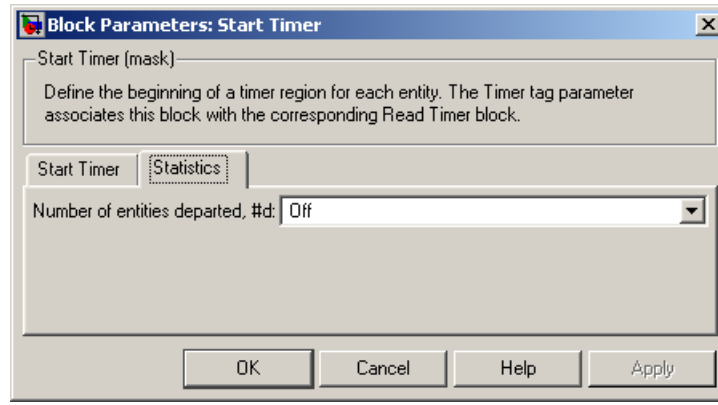
If timer has already started

Behavior of the block if an arriving entity already has a timer with the specified timer tag.

Statistics Tab

These parameters determine whether 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

Controls the presence and behavior of the signal output port labeled **#d**.

Examples

- "Basic Procedure for Using Timer Blocks" in the SimEvents user guide documentation
- "Timing Multiple Entity Paths with One Timer" in the SimEvents user guide documentation
- "Restarting a Timer from Zero" in the SimEvents user guide documentation
- "Timing Multiple Processes Independently" in the SimEvents user guide documentation

See Also

Read Timer

"Using Timers" in the SimEvents user guide documentation

Subsystem Configuration

Purpose

Configuration for Discrete Event Subsystem block

Library

SimEvents Ports and Subsystems

Description

Every Discrete Event Subsystem window must contain exactly one copy of this configuration block. Do not delete this block from the subsystem window or else the subsystem will not work properly. To create a subsystem using the Discrete Event Subsystem block, see "Setting Up Signal-Based Discrete Event Subsystems" in the SimEvents user guide documentation.

See Also

Discrete Event Subsystem, Discrete Event Inport

"Controlling Timing with Subsystems" in the SimEvents user guide documentation

Purpose

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 entities' generation events.

Intergeneration Times	Value of Generate entities with Parameter
Distributed according to various parameters in the block's 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`

For details about these options, see "Introduction to the Time-Based Entity Generator".

Responding to Blockage at the Entity Output Port

You can choose how this block responds when it generates an entity that the subsequent input port is not available to accept. If applicable, you can also choose how the block responds when that port later becomes available.

- If you set **Response when blocked** to `Error`, then the simulation halts with an error message if the subsequent entity input port is not available to accept an entity when this block generates it.
- If you set **Response when blocked** to `Pause generation`, then this block holds the generated entity and temporarily suspends the generation of additional entities. The **Response when unblocked** parameter determines when the held entity departs.

- If you set **Response when unblocked** to `Immediate restart`, then the held entity departs after this block receives notification that the subsequent entity input port has become available. After the held entity departs, this block schedules the generation of the next entity.
- If you set **Response when unblocked** to `Delayed restart`, then upon receiving notification that the subsequent entity input port has become available, this block schedules an extra generation event for the held entity. The event time is the current time plus the same intergeneration time used when first generating the held entity. When the extra generation event is processed, the held entity attempts to depart.

For an example, see "Example: Responding to Blockage" on page 4-260.

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 has departed and the block has updated 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 port appears only if you set **Generate entities with** to `Intergeneration time from port t`.

Entity Output Ports

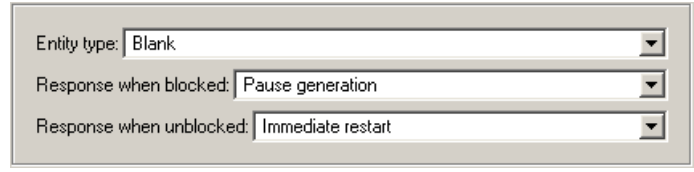
Label	Description
OUT	Port through which generated entities depart.

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 when the block is holding an entity that cannot depart because the subsequent entity input port is unavailable or because of a delayed restart of this block.	After entity generation if **OUT** port is blocked, and when entity departure is imminent in all cases	1
w	Average interdeparture time, in seconds, for all pairs of successive entities that have departed from this block. The signal value is 0 before the second entity departure.	After entity departure	2

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

Dialog Box

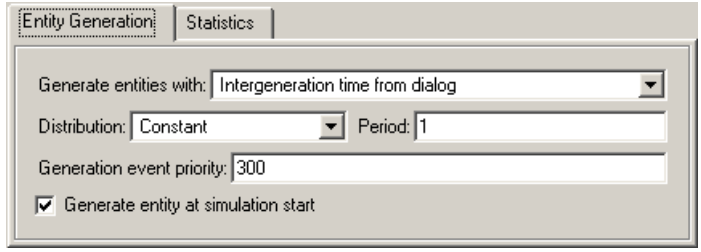


Entity type

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

Response when blocked, **Response when unblocked**
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 4-255.

Entity Generation Tab



Generate entities with
Determines where the block gets instructions about when to generate entities.

Distribution
The statistical distribution of intergeneration times. This field appears only if you set **Generate entities with** to `Intergeneration time from dialog`.

Period
The time interval between entity generations, in seconds. This field appears only if you set **Generate entities with** to `Intergeneration time from dialog` and set **Distribution** to `Constant`.

Initial seed
A nonnegative integer that initializes the random number generator. This field appears only if you set **Generate entities with** to `Intergeneration time from dialog` and set **Distribution** to `Uniform` or `Exponential`.

Minimum, **Maximum**
: The endpoints, in seconds, of the interval over which the distribution is uniform. These fields appear only if you set **Generate entities with** to `Intergeneration time from dialog` and set **Distribution** to `Uniform`.

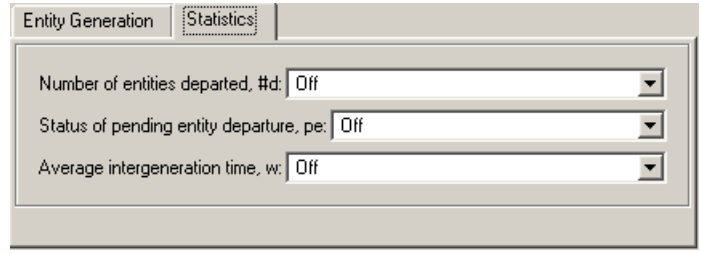
Mean
: The expected value of the exponential distribution. This field appears only if you set **Generate entities with** to `Intergeneration time from dialog` and set **Distribution** to `Exponential`.

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.

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
Controls the presence and behavior of the signal output port labeled **#d**.

Status of pending entity departure
Controls the presence and behavior of the signal output port labeled **pe**.

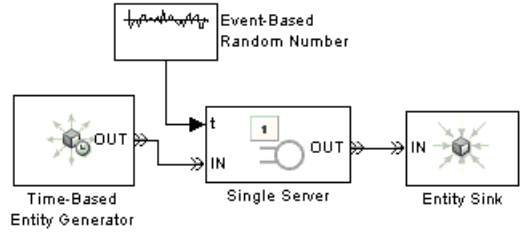
Average intergeneration time
Controls the presence and behavior of the signal output port labeled **w**.

Examples

- "Building a Simple Discrete-Event Model" in the SimEvents getting started documentation
- "Example: Using Random Intergeneration Times in a Queuing System" in the SimEvents getting started documentation
- "Example: Using a Step Function as Intergeneration Time" in the SimEvents getting started documentation
- "Example: Using an Arbitrary Discrete Distribution as Intergeneration Time" in the SimEvents getting started documentation
- "Specifying Generation Times for Entities" in the SimEvents user guide documentation

Example: Responding to Blockage

To illustrate the blockage options, consider a Time-Based Entity Generator block followed by a Single Server block, followed by an Entity Sink block.



Suppose the entity generator has **Response when blocked** set to `Pause generation`. Also, suppose that the intergeneration time is 1 with the first entity generated at T=1, and that the service times for the first three entities in the server are 1.5, 2.2, and 1.8. The tables below 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 server's entity input port becomes unavailable. The first entity is scheduled to depart from the server at 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 server's entity input port becomes available and the second entity advances from the entity generator to the server. The server's entity input port becomes unavailable again. The second entity is scheduled to complete service at T=2.5+2.2=4.7. The entity generator schedules the next generation for T=2.5+1=3.5.

Immediate Restart (Continued)

Time (s)	Behavior
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 server's entity input port becomes available and the third entity advances from the entity generator to the server. The server's entity input port becomes unavailable again. The third entity is scheduled to complete service at 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 and outputs the first entity to the server. The server's entity input port becomes unavailable. The first entity is scheduled to depart from the server at 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 server's entity input port becomes available. The entity generator schedules an extra generation 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 server's entity input port becomes unavailable again. The second entity is scheduled to complete service at T=3.5+2.2=5.7. The entity generator schedules the next generation for T=3.5+1=4.5.

Delayed Restart (Continued)

Time (s)	Behavior
4.5	Entity generator generates the third entity, and holds it because the **OUT** port is blocked.
5.7	Second entity departs from the server. The server's entity input port becomes available. The entity generator schedules an extra generation 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 server's entity input port becomes unavailable again. The third entity is scheduled to complete service at 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, Entity Sink

"Creating Entities Using Intergeneration Times" in the SimEvents getting started documentation

X-Y Attribute Scope

Purpose

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. For details, see "Connections Among Points in Plots" in the SimEvents user guide documentation.

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. This port appears only if you select **Enable entity OUT port**.

Signal Output Ports

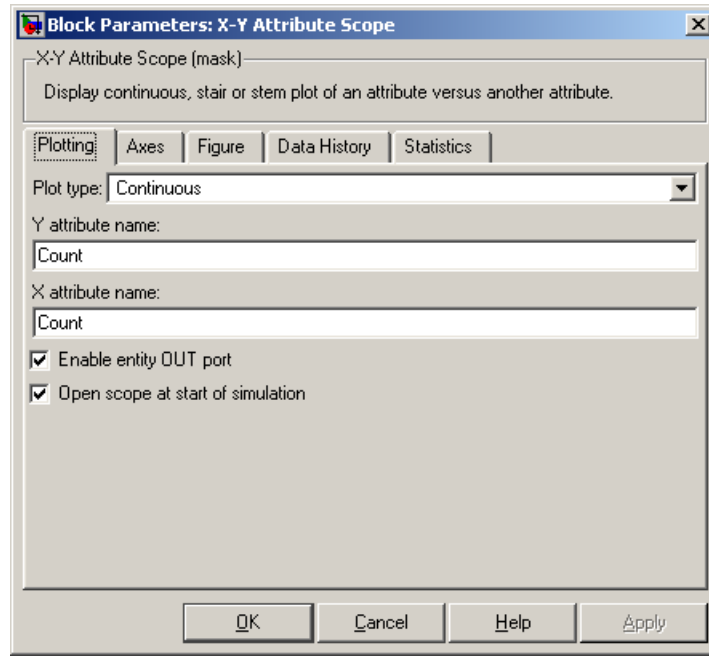
Label	Description
#a	Number of entities that have arrived at the block since the start of the simulation.

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

Dialog Box

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

Plotting Tab



Plot type

The presentation format for the data. See “Connections Among Points in Plots” in the SimEvents user guide documentation for details.

Y attribute name

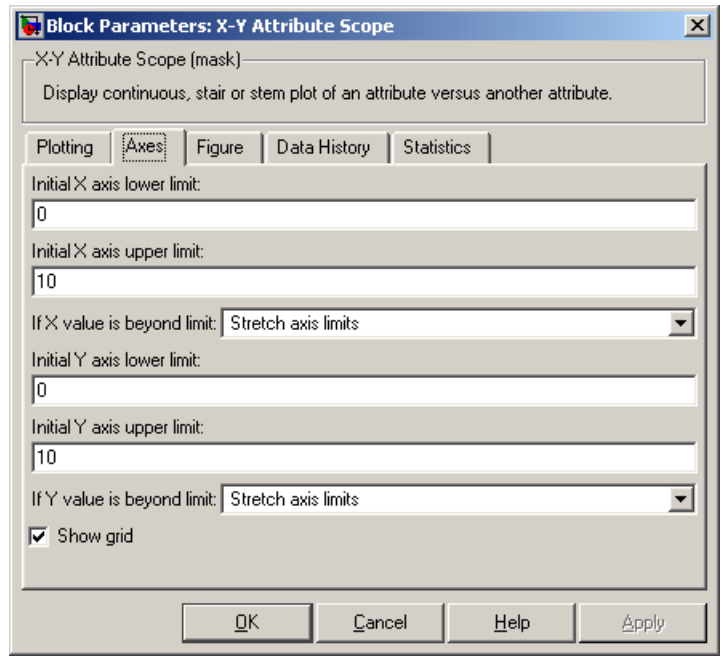
Name of the attribute to plot along the vertical axis.

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.

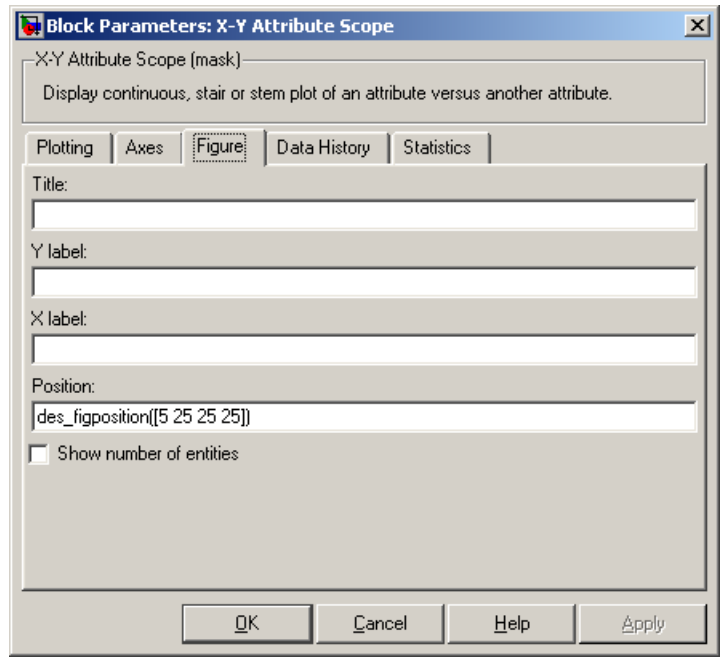
If X value is beyond limit
Determines how the plot changes if one or more X values are not within the limits shown on the X axis. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

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. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

Show grid
Toggles the grid on and off.

Figure Tab



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

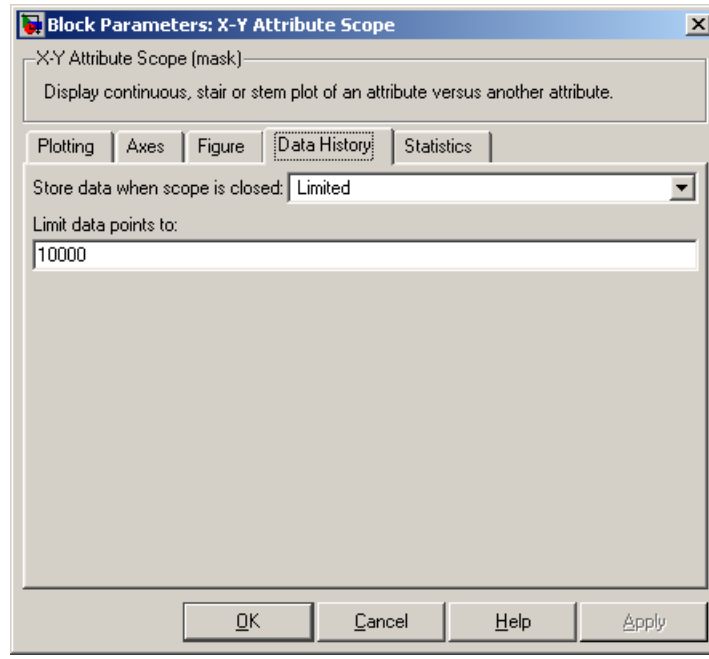
Y label
Text that appears to the left of the vertical axis.

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

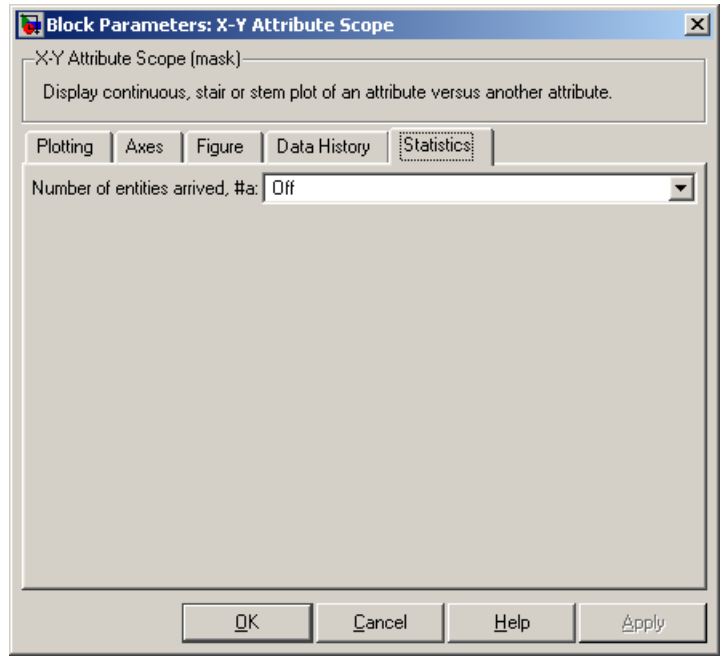


Store data when scope is closed
Select `Unlimited` to have the block cache all data for future viewing, `Limited` to cache a portion of the most recent data, and `Disabled` to avoid caching undisplayed data.

Limit data points to
The number of data points the block caches, using the most recent data. This field appears only if you set **Store data when scope is closed** to `Limited`.

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".



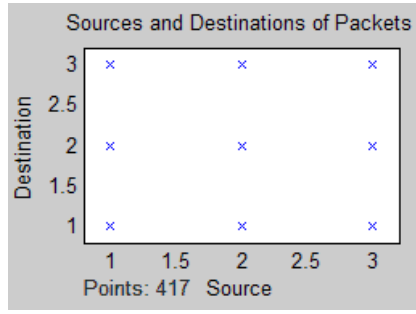
Number of entities arrived

Controls the presence and behavior of the signal output port labeled **#a**.

Examples

You can modify the example in "Example: A Packet Switch" in the SimEvents getting started documentation to check that all possible combinations of source and destination are used in the simulation. Insert an X-Y Attribute Scope block between the Path Combiner and

Output Switch blocks and use it to plot the `Destination` attribute against the `Source` attribute.



See Also

Attribute Scope, X-Y Signal Scope

"Plotting Data" in the SimEvents user guide documentation, "Accessing Attributes of Entities" in the SimEvents user guide documentation

X-Y Signal Scope

Purpose

Plot data from two signals

Library

SimEvents Sinks

Description

This block plots a curve using data from two input signals. The plot is particularly appropriate for data arising from discrete-event simulations or data related to entities because the plot can include zero-duration values.

The **Plot type** parameter on the **Plotting** tab determines whether and how the block connects the points that it plots. For details, see "Connections Among Points in Plots" in the SimEvents user guide documentation.

Ports

Signal Input Ports

Label	Description
in	Signal containing data for Y axis.
x	Signal containing data for X axis.

Signal Output Ports

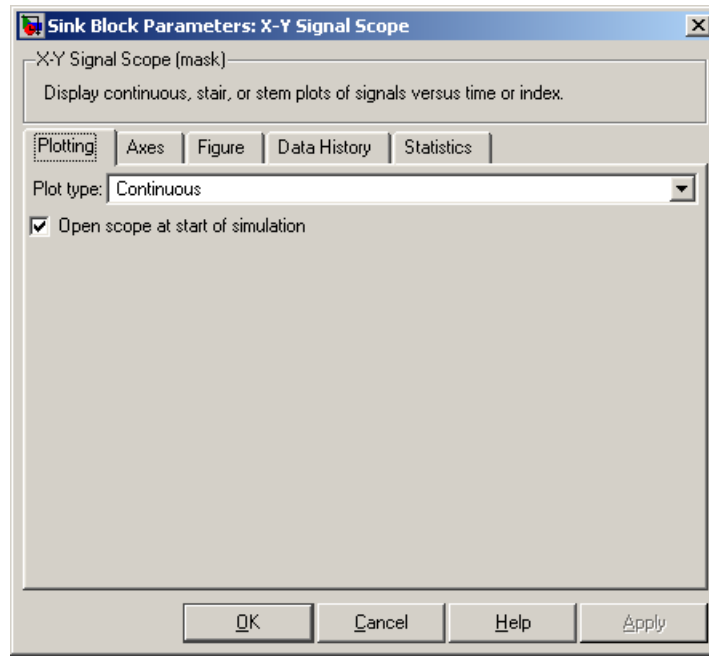
Label	Description
#c	Number of points the block has plotted.

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

Dialog Box

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

Plotting Tab



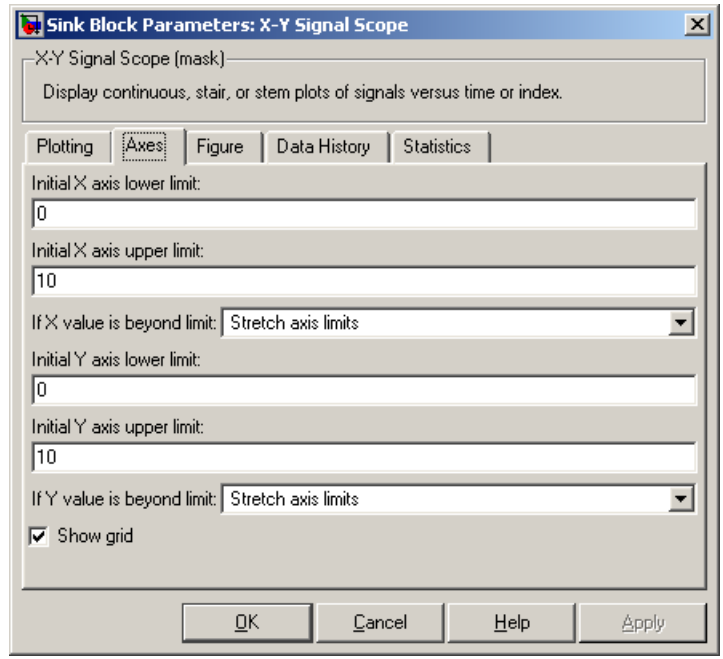
Plot type

The presentation format for the data. See “Connections Among Points in Plots” in the SimEvents user guide documentation 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.

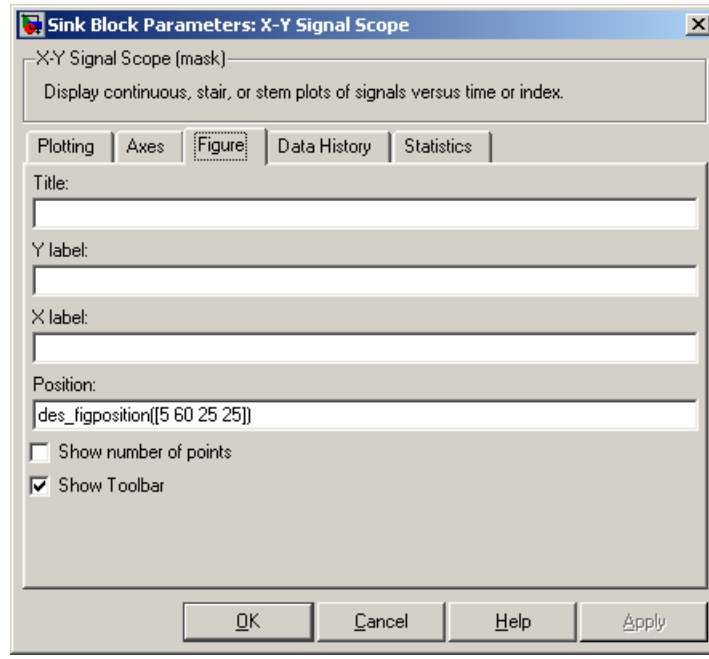
If X value is beyond limit
: Determines how the plot changes if one or more X values are not within the limits shown on the X axis. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

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. For details, see "Varying Axis Limits Automatically" in the SimEvents user guide documentation.

Show grid
Toggles the grid on and off.

Figure Tab



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

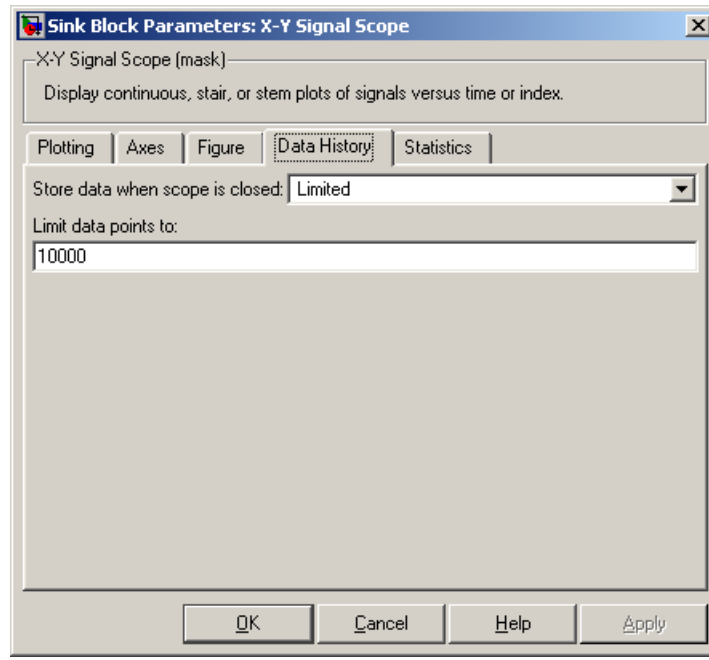
Y label
Text that appears to the left of the vertical axis.

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.

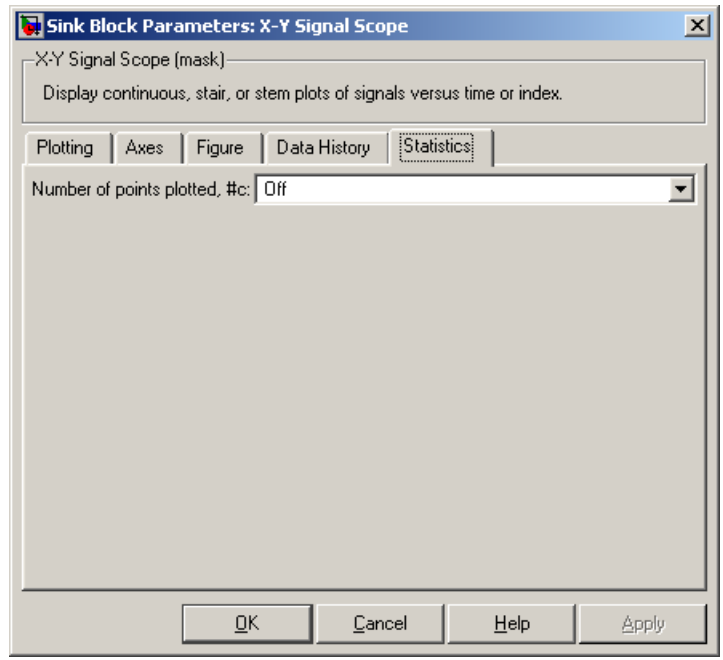
Limit data points to

The number of data points the block caches, using the most recent data. This field appears only if you set **Store data when scope is closed** to `Limited`.

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”.

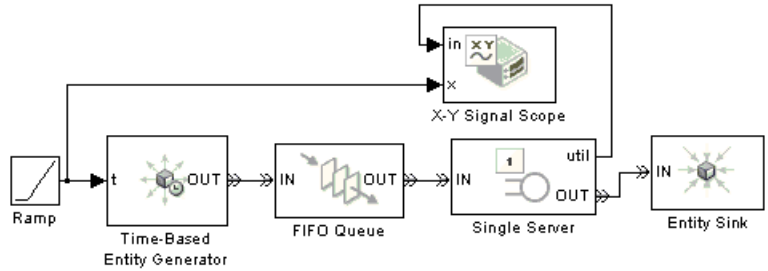


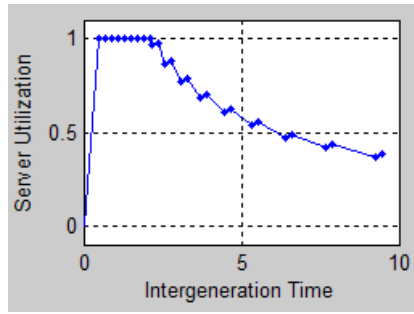
Number of points plotted

Controls the presence and behavior of 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, X-Y Attribute Scope

"Plotting Data" in the SimEvents user guide documentation

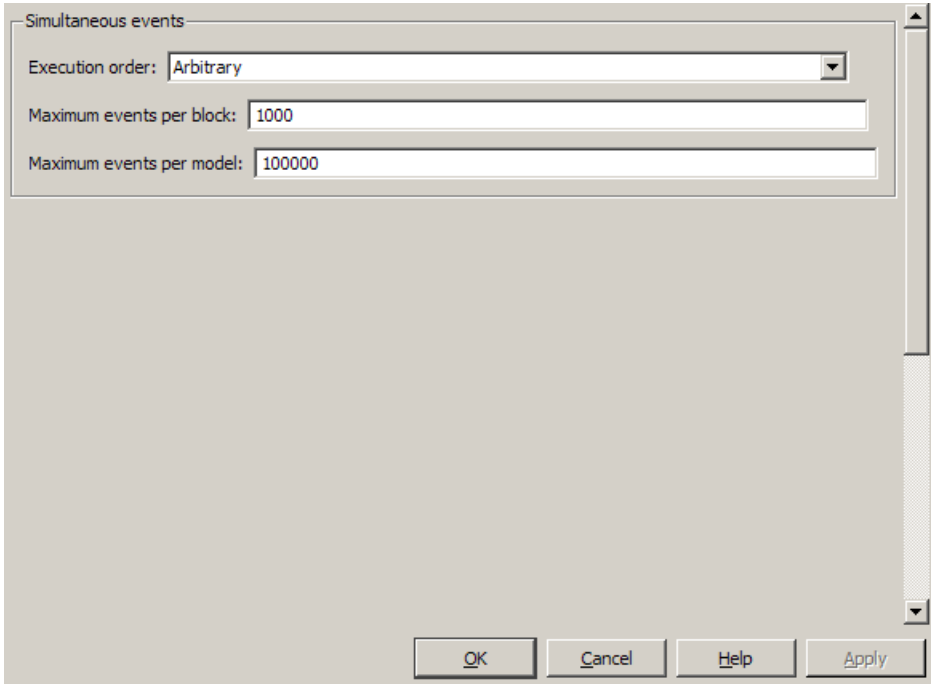
X-Y Signal Scope

5

Configuration Parameters

- "SimEvents Pane" on page 5-2
- "SimEvents Diagnostics Pane" on page 5-9

SimEvents Pane



In this section...
"SimEvents Pane Overview" on page 5-4
"Execution order" on page 5-5
"Seed for event randomization" on page 5-6
"Maximum events per block" on page 5-7
"Maximum events per model" on page 5-8

SimEvents Pane Overview

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

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`

See Also

- "Event Sequencing"
- "Procedure for Specifying Equal-Priority Behavior"

Seed for event randomization

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'`

See Also

"Unexpected Correlation of Random Processes"

Maximum events per block

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'`

See Also

"Livelock Prevention"

Maximum events per model

Limit the total number of events scheduled via the event calendar at each fixed time instant.

Settings

Default: `100000`

Minimum: `2`

Maximum: `2^31-1`

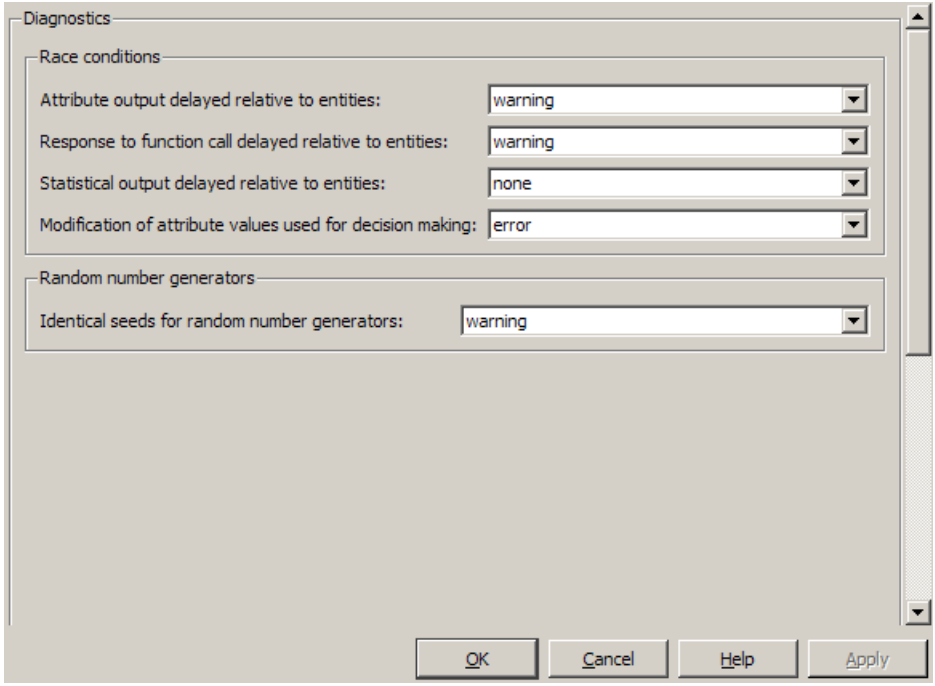
Command-Line Information

Parameter: `propMaxDesMdlSimulEvents`
Type: string
Value:
Default: `'100000'`

See Also

"Livelock Prevention"

SimEvents Diagnostics Pane



In this section...

"Diagnostics Pane Overview" on page 5-10

"Attribute output delayed relative to entities" on page 5-11

"Response to function call delayed relative to entities" on page 5-13

"Statistical output delayed relative to entities" on page 5-15

"Modification of attribute values used for decision making" on page 5-17

"Identical seeds for random number generators" on page 5-19

Diagnostics Pane Overview

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

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: `warning`

`none`
: The application does not check for this situation.

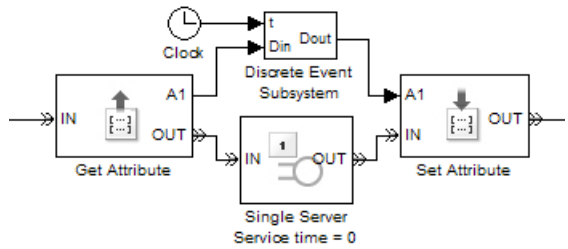
`warning`
: When the application detects this situation, it displays a warning.

`error`
: When the application detects this situation, it terminates the simulation and displays an error message.

Tip

A Single Server block whose **Service time** parameter is `0` can address the problem by storing the entity while the latter block responds to the signal update.

Example of Solution



For details, see "Using Block Diagrams to Manipulate Attributes".

Command-Line Information

Parameter: `propDiagAttribOutput`
Type: double
Value: `0 | 1 | 2`
Default: `1`

Recommended Settings

Application	Setting
Debugging	`warning` or `error`
Efficiency	`none`

See Also

- "Unexpected Use of Old Value of Signal"
- Managing Race Conditions demo

Response to function call delayed relative to entities

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: `warning`

`none`
: The application does not check for this situation.

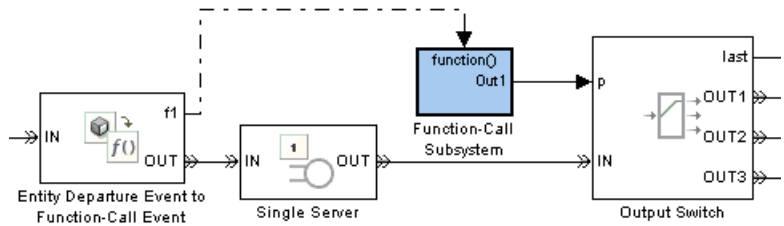
`warning`
: When the application detects this situation, it displays a warning.

`error`
: When the application detects this situation, it terminates the simulation and displays an error message.

Tip

A Single Server block whose **Service time** parameter is `0` can address the problem by storing the entity while subsequent blocks respond to the function call and its consequences.

Example of Solution



For details, see "Example: Using Entity-Based Timing for Choosing a Port".

Command-Line Information

Parameter: `propDiagFcnCallOutput`
Type: double
Value: `0 | 1 | 2`
Default: `1`

Recommended Settings

Application	Setting
Debugging	`warning` or `error`
Efficiency	`none`

See Also

"Unexpected Use of Old Value of Signal"

Statistical output delayed relative to entities

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: `warning`

`none`
: The application does not check for this situation.

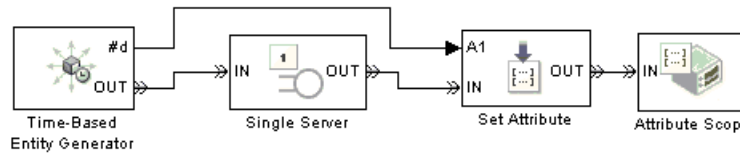
`warning`
: When the application detects this situation, it displays a warning.

`error`
: When the application detects this situation, it terminates the simulation and displays an error message.

Tip

A Single Server block whose **Service time** parameter is `0` can address the problem by storing the entity while the latter block responds to the signal update.

Example of Solution



For details, see "Example: Sequence of Departures and Statistical Updates".

Command-Line Information

Parameter: `propDiagStatOutput`

Type: double
Value: `0 | 1 | 2`
Default: `0`

Recommended Settings

Application	Setting
Debugging	`warning` or `error`
Efficiency	`none`

See Also

"Unexpected Use of Old Value of Signal"

Modification of attribute values used for decision making

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: `warning`

`none`
: The application does not check for this situation.

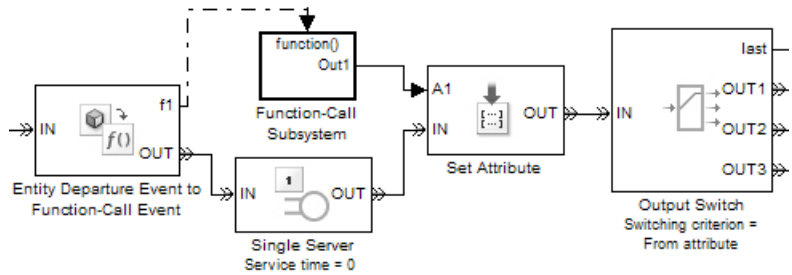
`warning`
: When the application detects this situation, it displays a warning.

`error`
: When the application detects this situation, it terminates the simulation and displays an error message.

Tip

A Single Server block whose **Service time** parameter is `0` can address the problem by storing the entity while the latter block responds to the signal update.

Example of Solution



The model is similar to the one in "Example: Using Entity-Based Timing for Choosing a Port", but performs switching based on an attribute rather than a signal value.

Command-Line Information

Parameter: `propDiagChangeAttrib`
Type: double
Value: `0 | 1 | 2`
Default: `2`

Recommended Settings

Application	Setting
Debugging	`warning` or `error`
Efficiency	`none`

See Also

"Querying Whether a Subsequent Block Can Accept an Entity"

Identical seeds for random number generators

Select the diagnostic action to take if the application detects that multiple random number generators use the same seed value, which might cause correlations among random processes.

Settings

Default: `warning`

`none`
: The application does not check for this situation.

`warning`
: When the application detects this situation, it displays a warning.

`error`
: When the application detects this situation, it terminates the simulation and displays an error message.

Tips

- If you set the parameter to `warning`, the warning message contains hyperlinks labeled “Randomize” and “Randomize All” that can help you address the problem.
- The `se_randomizeseeds` function provides a programmatic way to address the problem.
- Set the parameter to `none` if duplicate seeds are intentional in your model. For example, the Comparison of Routing Policies demo implements the same random arrival process multiple times, to compare routing policies.

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

- Avoiding Identical Seeds for Random Number Generators demo
- "Unexpected Correlation of Random Processes"
- `se_randomizeseeds`

Glossary

advance
To depart from one block and arrive immediately at another block. An entity advances from block to block during a simulation.

arrival
Entrance of an entity to a block via an entity input port. Arrival is the opposite of departure.

attribute
Data associated with an entity.

For example, an entity might be associated with a size, weight, speed, or part number.

available
The state of an entity input port that permits entities to arrive at the block.

For example, when a Single Server block is empty, its entity input port is available. When the block is busy serving, its entity input port is unavailable.

blocked
The state of an entity output port when an entity is trying to depart via the port and the port connects to an unavailable entity input port of another block.

For example, consider a FIFO Queue block whose entity output port is connected to the Single Server block's entity input port. Suppose the queue contains one entity. The queue's entity output port is blocked if the server's entity input port is unavailable, and not blocked if the server's entity input port is available. If the queue is empty, then its entity output port is not blocked because no entity is trying to depart.

component entity
An entity that forms part of a composite entity.

composite entity

An entity that comprises one or more entities as subordinate parts. The parts are called component entities.

departure

Exit of an entity from a block via an entity output port. Departure is the opposite of arrival.

discrete event subsystem

A subsystem containing time-based blocks that is called at the exact time of each qualifying event, rather than at times suggested by the time-based simulation clock. For the Discrete Event Subsystem block, qualifying events are signal-based events; for an appropriately configured Function-Call Subsystem block, qualifying events are function calls.

For example, the subsystem might contain blocks that end the simulation if the length of a queue exceeds 100, and might be configured so that the subsystem executes only at the exact moments when the queue reports an increased length.

entity

An abstract representation of an item of interest in a discrete-event simulation. The specific interpretation of an entity depends on what you are modeling. Entities can carry data, known as attributes.

For example, an entity could represent a packet in a communication network, a person using a bank of elevators, or a part on a conveyor belt.

entity input port

An input port at which an entity can potentially arrive. An entity input port can be available or unavailable; this state, which can change during the simulation, helps determine whether the port actually accepts the arrivals of new entities.

entity output port

An output port from which an entity can potentially depart. An entity output port can have a state of blocked or not blocked; this state, which can change during the simulation, determines whether the port's attempt to output an entity is successful.

entity path

A connection from an entity output port to an entity input port, depicted as a line connecting the entity ports of two blocks. An entity path represents the equivalence between an entity's departure from the first block and arrival at the second block. The connection line depicts a relationship between the two blocks.

An entity path is in active use by an entity only at zero or more discrete times during the simulation. By contrast, a connection line between signal ports represents a signal that has a well-defined value at all times during the simulation.

entity port

An entity input port or an entity output port.

entity priority

A real number associated with an entity, used to determine its sequence in a priority queue.

Contrast with event priority.

entity-departure subsystem

A kind of discrete event subsystem that is called at the exact time of each entity departure from a block or blocks, rather than at times suggested by the time-based simulation clock. See "Creating Entity-Departure Subsystems" in the SimEvents user guide documentation for details.

event

An instantaneous discrete incident that changes a state variable, an output, and/or the occurrence of other events. The prototypical events are arrivals and departures of entities. Examples of events are the generation of a new data packet in communications, the exit of a person from an elevator, and the placement of a new part on a conveyor belt.

event calendar

The internal list of events that are scheduled for the current time or future times.

For example, when a server begins its service time on a specific entity, the application inserts an entry into the event calendar for the completion of service on that entity at a future time. In a system

representing elevator passengers, this event calendar entry might represent the event whereby a specific person in an elevator reaches the desired floor.

event priority
A positive integer associated with an event scheduled on the event calendar, used to sequence the processing of simultaneous events. Simultaneous events having distinct numerical event priorities are processed in ascending order of the event priority values.

Contrast with entity priority.

event translation
Conversion of one event into another. The result of the translation is often a function call, but can be another type of event. The result of the translation can occur at the same time as, or a later time than, the original event.

event-based signal
A signal that can change in response to discrete events. For example, the signal representing the number of entities in a queue changes upon each arrival at or departure from the queue.

event-based simulation
A simulation that permits the system's state transitions to depend on asynchronous discrete incidents called events.

function-call event
A discrete invocation request carried from block to block by a special signal called a function-call signal. A function-call event is also called simply a function call.

intergeneration time
The time interval between successive generations, as in a Time-Based Entity Generator block.

monitoring port
A signal input port that is designed for observing signal values. Contrast with notifying port.

notifying port

A signal input port that notifies the preceding block when a certain event has occurred. When the preceding block is the Event-Based Random Number block, it responds to the notification by generating a new random number.

For example, the **t** input port of a Single Server block is a notifying port; when connected to this port, the Event-Based Random Number block generates a new random number each time it receives notification that an entity has arrived at the server.

pending entity

An entity that tries and fails to depart from a block. The failure occurs because the entity output port through which the entity would depart is connected to an unavailable entity input port of another block.

preemption

The replacement of an entity in a server block by an entity that satisfies certain criteria.

reactive port

A signal input port that listens for updates or changes in the input signal and causes an appropriate reaction in the block possessing the port. For example, the **p** port on an Input Switch block listens for changes in the input signal; the block reacts by selecting a new entity port for potential arrivals.

sample time hit event

An update in the value of a signal that is connected to a block configured to react to signal updates. The updated value could be the same as or different from the previous value.

signal port

An input or output port that represents a numerical quantity that changes over time and that is defined for all times during the simulation. Unlike an entity port, a signal port has no state and does not have entity arrivals or entity departures.

signal-based event

A sample time hit event, value change event, or trigger event.

simultaneous events

Events that occur at the same value, or sufficiently close values, of the simulation clock. Events scheduled on the event calendar for times T and $T+\Delta t$ are considered simultaneous if $0 \le \Delta t \le$ `128*eps*`T, where `eps` is the floating-point relative accuracy in MATLAB software and T is the simulation time.

For example, in a D/D/1 queuing system where the arrival rate equals the service rate, an entity generation event and a service completion event are simultaneous. Parameters in the model determine which of these events occurs first, though the clock has the same value in both cases.

time-based signal

A signal that can change only in response to the simulation clock.

time-based simulation

A simulation in which state transitions depend on time.

For example, a simulation based solely on differential equations in which time is an independent variable is a time-based simulation.

timeout event

An event that causes an entity to depart via a special output port when the entity has exceeded a previously established time limit. Use the Schedule Timeout block to schedule a timeout event for each arriving entity. Optionally, use the Cancel Timeout block to remove a timeout event from the event calendar before the event occurs. See "Forcing Departures Using Timeouts" in the SimEvents user guide documentation for details.

timeout interval

The duration between an entity's arrival at a Schedule Timeout block and the scheduled time of the entity's timeout event. See "Forcing Departures Using Timeouts" in the SimEvents user guide documentation for details.

trigger edge

A rising edge or falling edge of a signal. A rising edge is an increase from a negative or zero value to a positive value (or zero if the initial

value is negative). A falling edge is a decrease from a positive or a zero value to a negative value (or zero if the initial value is positive).

trigger event
A trigger edge in a signal that is connected to a block configured to react to trigger edges.

trigger signal
A signal whose trigger edges are used to invoke a behavior during the simulation.

unavailable
The state of an entity input port that prevents entities from arriving at the block.

For example, when a Single Server block is empty, its entity input port is available. When the block is busy serving, its entity input port is unavailable.

value change event
An increase or decrease in the numerical value of a signal that is connected to a block configured to react to relevant changes.

zero-duration value
A value that an event-based signal assumes at an instant in time but that does not persist for a positive duration.

For example, when a full N-server advances one entity to the next block, the statistical signal representing the number of entities in the block assumes the value N-1. However, if the departure causes another entity to arrive at the block at the same time instant, then the statistical signal assumes the value N. The value of N-1, which does not persist for a positive duration, is a zero-duration value. This phenomenon occurs in many situations; see “Working with Multivalued Signals” in the SimEvents user guide documentation for details.

Index

A

Attribute Function block 4-2
Attribute Scope block 4-5

B

`bdelete` function 2-10
`blkinfo` function 2-13
`blklist` function 2-20
`breakpoints` function 2-22

C

Cancel Timeout block 4-15
configuration parameters 5-4
    SimEvents Diagnostics pane 5-10
        Attribute output delayed relative to entities 5-11
        Identical seeds for random number generators 5-19
        Modification of attribute values used for decision making 5-17
        Response to function call delayed relative to entities 5-13
        Statistical output delayed relative to entities 5-15
    SimEvents® pane
        Execution order 5-5
        Maximum events per block 5-7
        Maximum events per model 5-8
        Seed for event randomization 5-6
Conn block 4-20
`cont` function 2-26
`currentop` function 2-28

D

delays
    entities 4-104
`detail` function 2-31
`disable` function 2-36
Discrete Event Inport block 4-22
Discrete Event Outport block 4-25
Discrete Event Signal to Workspace block 4-26
Discrete Event Subsystem block 4-29

E

`enable` function 2-39
Enabled Gate block 4-32
`eninfo` function 2-42
Entity Combiner block 4-36
Entity Departure Counter block 4-43
Entity Departure Event to Function-Call Event block 4-47
Entity Sink block 4-53
Entity Splitter block 4-55
Entity-Based Function-Call Event Generator block 4-63
`evbreak` function 2-46
`evcal` function 2-50
Event-Based Entity Generator block 4-66
Event-Based Random Number block 4-72
Event-Based Sequence block 4-83
`evinfo` function 2-55

F

FIFO Queue block 4-87

G

`gceb` function 2-57
`gcebid` function 2-61
`gcen` function 2-65
`gcev` function 2-69
Get Attribute block 4-94

H

`help` function 2-2

I

Infinite Server block 4-104
Initial Value block 4-111
Input Switch block 4-114
Instantaneous Entity Counting Scope block 4-119
Instantaneous Event Counting Scope block 4-125

L

LIFO Queue block 4-132

N

N-Server block 4-138

O

Output Switch block 4-146

P

Path Combiner block 4-158
ports
event-based random numbers 4-72
event-based signals 4-84
Priority Queue block 4-165

Q

`quit` function 2-72

R

Read Timer block 4-172
Release Gate block 4-177
Replicate block 4-182
`runtoend` function 2-73

S

Schedule Timeout block 4-189
`se_getdbopts` function 2-3
`se_getseeds` function 2-5
`se_randomizeseeds` function 2-6
`se_setseeds` function 2-8
`sedb.bdelete` function 2-10
`sedb.blkinfo` function 2-13
`sedb.blklist` function 2-20
`sedb.breakpoints` function 2-22
`sedb.cont` function 2-26
`sedb.currentop` function 2-28
`sedb.detail` function 2-31
`sedb.disable` function 2-36
`sedb.enable` function 2-39
`sedb.eninfo` function 2-42
`sedb.evbreak` function 2-46
`sedb.evcal` function 2-50
`sedb.evinfo` function 2-55
`sedb.gceb` function 2-57
`sedb.gcebid` function 2-61
`sedb.gcen` function 2-65
`sedb.gcev` function 2-69
`sedb.quit` function 2-72
`sedb.runtoend` function 2-73
`sedb.simtime` function 2-74
`sedb.step` function 2-76
`sedb.tbreak` function 2-78
`sedebug` function 2-81
servers
infinite 4-104
Set Attribute block 4-195
Signal Latch block 4-207
Signal Scope block 4-218
Signal-Based Event to Function-Call Event block 4-227
Signal-Based Function-Call Event Generator block 4-234
`simeventsconfig` function 2-83
`simeventsdocex` function 2-85

simeventslib function 2-86
simeventsstartup function 2-87
simtime function 2-74
Single Server block 4-238
Start Timer block 4-251
step function 2-76
Subsystem Configuration block 4-254

T

tbreak function 2-78
Time-Based Entity Generator block 4-255

X

X-Y Attribute Scope block 4-264
X-Y Signal Scope block 4-272