

SimEvents[®]

Reference



MATLAB[®]&SIMULINK[®]

R2015a



How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



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

SimEvents[®] Reference

© COPYRIGHT 2005–2015 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

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

Alphabetical List

1

Blocks — Alphabetical List

2

Configuration Parameters

3

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

Upgrade Advisor Checks

4

SimEvents Upgrade Advisor Checks	4-2
Checks Overview	4-2
Check model and local libraries for legacy SimEvents blocks .	4-3
Check for implicit event duplication caused by SimEvents blocks	4-4

SimEvents Terminology

Alphabetical List

help

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.

Input Arguments

functionname

Name of a SimEvents debugger function or other function.

More About

- “Start the SimEvents Debugger”

se_getdbopts

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 that the debugger executes after the debugger session begins.

Output Arguments

`opts_struct`

Structure that describes options for a SimEvents debugger session. The `StartFcn` field of the structure is a cell array of strings.

Examples

Set breakpoints upon starting the debugger:

- 1 At the MATLAB command prompt, create an options structure to 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 execute the commands in the `opts_struct.StartFcn` field automatically:

```
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 | Help | Watch Video Tutorial  
  
%=====  
Initializing Model sedemo_outputswitch  
Set b1 : Breakpoint for first operation at or after time 1  
Set b2 : Breakpoint for first operation at or after time 2
```

- 3 Proceed in the simulation until the first breakpoint. At the `sedebg>>` prompt, enter:

```
cont
```

The output is:

```
%=====  
Initializing Time-Based Entity Generators  
%.....%  
Scheduling EntityGeneration Event (ev1)  
: EventTime = 1.0000000000000000  
: Priority = 500  
: Entity = <none>  
: Block = Time-Based Entity Generator  
  
Hit b1 : Breakpoint for first operation at or after time 1  
  
%=====  
Executing EntityGeneration Event (ev1)           Time = 1.0000000000000000  
: Entity = <none>                               Priority = 500  
: Block = Time-Based Entity Generator
```

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

```
sedb.quit
```

More About

- “Debugger Efficiency Tips”

See Also

sedebg

se_getseeds

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. `sysid` indicates the scope of the search by specifying a system name, subsystem path name, or block path name.

Input Arguments

sysid

String that indicates the scope of the search by specifying a system name, subsystem path name, or block path name.

Output Arguments

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

More About

Tips

- Before invoking this function, load or open the system.
- “Make Results Repeatable by Storing Sets of Seeds”
- “Share Seeds Among Models”

See Also

`se_setseeds` | `se_randomizeseeds`

Tutorials

- Seed Management Workflow for Random Number Generators

se_randomizeseeds

Randomize seeds

Syntax

```
se_randomizeseeds(obj,Name,Value)
se_randomizeseeds(obj,'Mode','All',Name,Value)
se_randomizeseeds(obj,'Mode','Identical',Name,Value)
se_randomizeseeds(obj,'Mode','SpecifySeeds',sv,Name,Value)
```

Description

`se_randomizeseeds(obj,Name,Value)` or `se_randomizeseeds(obj,'Mode','All',Name,Value)` assigns the seeds of all SimEvents blocks that use random number generators within blocks indicated by `obj`. If `obj` represents a system or subsystem, the function assigns seeds in subsystems of `obj` at any depth. Zero or more `Name,Value` pair arguments specify additional options. Unless the syntax includes the `GlobalSeed` option, the new seeds are unique in the system `obj`.

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

`se_randomizeseeds(obj,'Mode','SpecifySeeds',sv,Name,Value)` assigns only those seeds whose current value appears in the vector `sv`.

Input Arguments

obj

Location of seeds to assign. `obj` is either a string representing a system name, subsystem path name, or block path name, or a cell array of such strings.

sv

Vector of seed values. Each number is an integer between 0 and $2^{31}-1$.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (' '). You can specify several name and value pair arguments in any order as `Name1, Value1, . . . , NameN, ValueN`.

'GlobalSeed'

For a given value, the function generates seeds in a repeatable way, assuming that the underlying systems or blocks specified in `obj` do not change. To ensure repeatability, this syntax does not guarantee uniqueness of generated seed values. The value of this option is a nonnegative integer.

'Verbose'

Indicates whether the function explicitly reports the status of each seed assignment. Values are 'on' and 'off'.

Default: 'off'

More About

Tips

- Before invoking this function, load or open the system where you want to assign seeds.

See Also

`se_getseeds` | `se_setseeds`

Tutorials

- [Avoiding Identical Seeds for Random Number Generators](#)
- [Seed Management Workflow for Random Number Generators](#)

se_setseeds

Set seed values for blocks with random number generators

Syntax

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

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

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

Input Arguments

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.

Output Arguments

oldseedstruct

Structure representing original seed values. The form of `oldseedstruct` is the same as that of `seedstruct`.

status

Logical array whose *n*th entry is false if the *n*th block in `seedstruct` meets one of these conditions:

- Does not exist
- Is not a SimEvents block
- Does not have a seed parameter in its current configuration

More About

Tips

- Before invoking this function, load or open the system where you want to set seeds.
- “Set Seed Values Programmatically”
- “Share Seeds Among Models”

See Also

`se_getseeds` | `se_randomizeseeds`

Tutorials

- Seed Management Workflow for Random Number Generators

sedb.animate

Package: sedb

Turn on or off animation and control speed

Syntax

```
animate  
animate on  
animate off  
animate on T  
animate T  
animate(structure)  
previous = animate(...)
```

Description

`animate` turns the animation capability on and off. By default, animation is on when you start debugging the model.

`animate on` turns the animation capability on.

`animate off` turns the animation capability off.

`animate on T` and `animate T` sets a delay for the animation. The delay controls the speed of the animation.

`animate(structure)` sets the animation settings to those specified in *structure*.

`previous = animate(...)` returns the previous state of the animation as a structure.

Input Arguments

T

Positive value between 0 and 5 that specifies the animation delay. This value controls the animation speed.

Default: 0

structure

List of options that you specify for `animate`. This structure has the following fields:

Field	Description
state	Contains animation state, 'on' or 'off'.
delay	Contains animation delay, from real value from 0 to 5.

Output Arguments

previous

Structure of previously specified animation settings. This structure has the following fields:

Field	Description
state	Contains animation state, 'on' or 'off'
delay	Contains animation delay, from real value from 0 to 5

Examples

Turn animation on with delay 0.5 sec.

```
animate on 0.5
```

Return previously set animation settings in *previous*.

```
previous=animate
```

Specify animation settings in structure *x*, and return them in structure *y*.

```
x=animate(0.1);
y=animate(x);
```

More About

Tips

- Use the `detail` command to control the model components that the animation displays.
- A negative delay value stops the animation.
- “Debug Models Using Animation”

See Also

sedb.bdelete

Package: sedb

Delete breakpoints in discrete-event simulation

Syntax

```
bdelete id1 id2 id3 ...  
bdelete(id_array)  
bdelete all
```

Description

`bdelete id1 id2 id3 ...` deletes the breakpoint having identifier `id`. 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.

Input Arguments

id

String that represents a breakpoint identifier.

id_array

Cell array of strings that represent breakpoint identifiers.

Examples

Manipulate timed breakpoints and run the simulation until a breakpoint:

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

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

- 2 Establish and view breakpoints. At the prompt, enter:

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

The output lists the breakpoints:

```
Set b1 : Breakpoint for first operation at or after time 0.5  
Set b2 : Breakpoint for first operation at or after time 1  
Set b3 : Breakpoint for first operation at or after time 1.5
```

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  
Disabled b3 : Breakpoint for first operation at or after time 1.5  
Disabled b1 : Breakpoint for first operation at or after time 0.5  
Enabled b3 : Breakpoint for first operation at or after time 1.5
```

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 1.5

```
%=====
Executing EntityGeneration Event (ev7)           Time = 1.5000000000000000
: Entity = <none>                               Priority = 1
: Block = Time-Based Entity Generator
```

- 5 End the debugger session. At the prompt, enter:

```
sedb.quit
```

More About

- “Use Breakpoints During Debugging”

See Also

`sedb.breakpoints` | `sedb.disable`

sedb.blkbreak

Package: sedb

Set breakpoint for discrete-event simulation block

Syntax

```
blkbreak(blkid)
blkbreak(blkname)
bid = blkbreak(blkid)
bid = blkbreak(blkname)
```

Description

`blkbreak(blkid)` sets a breakpoint for the `SimEvents` block with identifier `blkid`. To obtain a list of blocks and their identifiers, use `sedb.blklist`.

`blkbreak(blkname)` sets a breakpoint for the `SimEvents` block with path name `blkname`.

`bid = blkbreak(blkid)` or `bid = blkbreak(blkname)` returns the identifier of the breakpoint.

The following blocks are exceptions that do not support block breakpoints:

- Conn
- Event-Based Random Number
- Event-Based Sequence
- Initial Value

Input Arguments

blkid

String that represents an identifier of a `SimEvents` block.

blkname

String that represents the path name of a SimEvents block.

Output Arguments

bid

String that represents a breakpoint identifier.

Examples

Examine a routing decision using a breakpoint on an Output Switch block:

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

```
sedebg('sedemo_arq_selective_repeat')
```

- 2 Double-click the Receiver subsystem to open it. On the Output Switch block inside the Receiver subsystem, establish a breakpoint. At the `sedebg>>` prompt, enter:

```
blkbreak('sedemo_arq_selective_repeat/Receiver/Output Switch')
```

The output confirms the creation of the block breakpoint:

```
Set b1 : Breakpoint for block (blk6) sedemo_arq_selective_repeat/Receiver/Output Switch
```

- 3 Proceed until the simulation reaches the breakpoint:

```
cont
```

The end of the partial output indicates that a sample time hit has been detected:

```
%.....%
Entity Advancing (en2)
: From = Forward Channel/Infinite Server
: To = Addition of Errors/Set Attribute
%.....%
Setting Attribute on Entity (en2)
: crc_check = 0
: Block = Addition of Errors/Set Attribute
%.....%
Entity Advancing (en2)
: From = Addition of Errors/Set Attribute
```

```

: To = Receiver/Get Attribute
%.....%
Entity Advancing (en2)
: From = Receiver/Get Attribute
: To = Receiver/Zero Delay
%.....%
Scheduling ServiceCompletion Event (ev7)
: EventTime = 1.100000000000000 (Now)
: Priority = 10
: Entity = en2
: Block = Receiver/Zero Delay
%.....%
Scheduling Subsystem Event (ev8)
: EventTime = 1.100000000000000 (Now)
: Priority = 1
: Entity = <none>
: Block = Receiver/Discrete Event Subsystem/tp63efdc8c_2b3b_442f_9f27_f5
%=====
Executing Subsystem Event (ev8)                               Time = 1.100000000000000
: Entity = <none>                                             Priority = 1
: Block = Receiver/Discrete Event Subsystem/tp63efdc8c_2b3b_442f_9f27_f5

Hit b1 : Breakpoint for block (blk6) sedemo_arq_selective_repeat/Receiver/Output Switch

%.....%
Detected Sample Time Hit
: Block = Receiver/Output Switch

```

- 4 Examine the source of the new value, 1. In the Receiver subsystem, the blocks and signal connections indicate that the **p** signal arises from a computation in the Discrete Event Subsystem virtual subsystem. The computation involves an attribute of an entity. The Get Attribute block dialog box indicates that the name of the attribute is `crc_check`. To determine which entity has the attribute in this particular computation, use the output from the preceding step. An entity with identifier `en2` departed from the Get Attribute block. Inspect the attributes of this entity, as follows:

```
eninfo en2
```

The output shows a `crc_check` value of 0. According to the main description of the model, a CRC value of 0 indicates an error.

```

Entity (en2) Current State                               T = 1.100000000000000
Location: Receiver/Zero Delay

Attributes:
  Name      Value
  crc_check 0
  seqNum    1

Timeouts, Timers: None

```

- 5 Proceed in the simulation to the next operation of the switch block:

```
cont
```

The switch block is about to respond to the new value of the **p** input signal by scheduling a port selection event:

```
Hit b1 : Breakpoint for block (blk6) sedemo_arq_selective_repeat/Receiver/Output Switch

%.....%
Scheduling PortSelection Event (ev9)
: EventTime = 1.1000000000000000 (Now)
: Priority   = SYS1
: Entity    = <none>
: Block     = Receiver/Output Switch
```

6 Proceed further in the simulation to the next operation:

cont

The output shows that the switch block is about to execute the port selection event. Executing that event causes the switch to select the entity output port corresponding to the new value of the **p** input signal.

```
Hit b1 : Breakpoint for block (blk6) sedemo_arq_selective_repeat/Receiver/Output Switch

%=====
Executing PortSelection Event (ev9)           Time = 1.1000000000000000
: Entity = <none>                           Priority = SYS1
: Block  = Receiver/Output Switch
```

7 Continue to proceed further in the simulation to the next operation:

cont

The output shows that the entity with identifier **en2** is about to advance to the switch block:

```
%=====
Executing ServiceCompletion Event (ev7)       Time = 1.1000000000000000
: Entity = en2                               Priority = 10
: Block  = Receiver/Zero Delay

%.....%
Entity Advancing (en2)
: From = Receiver/Zero Delay
: To   = Receiver/Entity Departure Event to Function-Call Event

Hit b1 : Breakpoint for block (blk6) sedemo_arq_selective_repeat/Receiver/Output Switch

%.....%
Entity Advancing (en2)
: From = Receiver/Entity Departure Event to Function-Call Event
: To   = Receiver/Output Switch
```

8 Continue to proceed further in the simulation to the next operation:

cont

The output shows that the entity advances to the Discarded Frames block, consistent with a CRC value of 0, indicating an error:

```
Hit b1 : Breakpoint for block (blk6) sedemo_arq_selective_repeat/Receiver/Output Switch
%.....%
Entity Advancing (en2)
: From = Receiver/Output Switch
: To   = Receiver/Discarded Frames
```

- 9 Disable the breakpoint for the Output Switch block by referencing its breakpoint identifier:

```
disable b1
```

Disabling a block breakpoint, when you are done investigating the block, helps you focus on operations that are relevant to you. The output confirms the disabling of the block breakpoint:

```
Disabled b1 : Breakpoint for block (blk6) sedemo_arq_selective_repeat/Receiver/Output Switch
```

- 10 End the debugger session. At the `sedebg>>` prompt, enter:

```
sedb.quit
```

More About

- “Define a Breakpoint”

See Also

`sedb.bdelete` | `sedb.blklist` | `sedb.breakpoints` | `sedb.cont`

sedb.blkinfo

Package: sedb

Block information in discrete-event simulation

Syntax

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

Description

`blkinfo(blkid)` displays information about the `SimEvents` block with identifier `blkid`. To obtain a list of blocks and their identifiers, use `sedb.blklist`.

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

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

The following blocks are exceptions that do not provide information:

- Conn
- Event-Based Random Number
- Event-Based Sequence
- Initial Value

Input Arguments

blkid

String that represents an identifier of a `SimEvents` block.

blkname

String that represents the path name of a SimEvents block.

Output Arguments

blk_struct

Structure that stores information about the block. The following table describes the blk_struct fields and the blocks for which each field appears. Some fields appear for all blocks, while other fields appear only for certain blocks.

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 that the block is replicating
CurrentReplica	Replicate	Identifier of the copy of the input entity that the block created and that is about to advance
Status	Replicate, Entity Combiner, and Entity Splitter	Status of block Values for Replicate Block: <ul style="list-style-type: none"> • 'Inactive' • 'Replicating X/Y'. X is the index of the entity output port through which the replica will depart. Y is the Number of

Field	Block	Description
		<p>entity output ports parameter of the block</p> <p>Values for Entity Combiner Block:</p> <ul style="list-style-type: none"> • 'Inactive' • 'Combining' <p>Values for Entity Splitter Block:</p> <ul style="list-style-type: none"> • 'Inactive' • 'Splitting'
MemoryValue	Signal Latch	Value of the internal memory of the block
Count	Entity Departure Counter	Number of entities that have departed from this block and arrived at subsequent storage blocks, since the simulation start or the last reset, whichever occurred later.

Field	Block	Description
Entities	All blocks that possess an entity input port	<p>Structure array, of which each element has these fields:</p> <ul style="list-style-type: none"> • ID — Entity identifier • Status (queue and server blocks only) — Status of the entity with respect to the block. <p>Entity Status Values in Queue Blocks:</p> <ul style="list-style-type: none"> • 'Advancing' • 'Queuing' • 'Queued' • 'Timed Out' <p>Entity Status Values in Server Blocks:</p> <ul style="list-style-type: none"> • 'Advancing' • 'In Service' • 'Service Completed' • 'Preempted' • 'Timed Out' <ul style="list-style-type: none"> • 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

Examples

View information about a block:

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

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

- 2 Establish a breakpoint and proceed in the simulation. At the prompt, enter:

```
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 2.51

%=====
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 =
blk4
```

- 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 display information about the block and the status of entities in it:

```
% Display information in Command Window.
blkinfo(blkid)
```

The output is:

```
InfiniteServer Current State          T = 2.581301297500028
Block (blk4): 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
```

- 6 Store the information in variables:

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

The output is:

```
blkdetails =
    Time: 2.5813
    Block: 'sedemo_start_timer_read_timer/Infinite Server'
    BlockID: 'blk4'
    BlockType: 'InfiniteServer'
    Capacity: Inf
    Entities: [1x3 struct]

blkentities =
    'In Service'
    'Advancing'
    'In Service'
```

- 7 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

8 End the debugger session. At the prompt, enter:

```
sedb.quit
```

More About

- “Inspect Entities, Blocks, and Events”

See Also

`sedb.blkbreak` | `sedb.blklist` | `sedb.gceb`

sedb.blklist

Package: sedb

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 that you are debugging. The list includes all blocks for which `blkinfo` or `blkbreak` is valid, including controlled Simulink® blocks and the gateway blocks. The list excludes virtual subsystems but includes relevant blocks inside virtual subsystems. In the Command Window, you can click hyperlinks to highlight the blocks in the model window.

`blk_cell = blklist` returns the same information as `blklist` in a cell array of strings.

Output Arguments

blk_cell

Cell array of strings. The first column of `blk_cell` contains block identifiers. In the second column, the corresponding cells contain block path names.

Examples

In the Command Window, view the block list and store it in a variable:

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

```
sedbug('sedemo_count_attributes')
```

- 2 View the list of blocks. At the `sedebg>>` prompt, enter:

```
blklist
```

The output is:

```
List of blocks in model: sedemo_count_attributes
```

```
blk1          Entity Data
blk3          Entity Sink
blk6          Event-Based Random Number
blk2          Get Attribute
blk4          Set Attribute
blk5          Time-Based Entity Generator
```

- 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 Data
```

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

```
sedb.quit
```

More About

- “Inspect Blocks”

See Also

[sedb.blkbreak](#) | [sedb.blkinfo](#)

sedb.breakpoints

Package: sedb

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.

`b_struct = breakpoints` returns a structure array that stores information about breakpoints in the simulation.

Output Arguments

b_struct

Structure array that stores information about breakpoints. The following table describes the fields of each structure in the array.

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

Set and view breakpoints:

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

```
sedebg('sedemo_timeout')
```

- 2 Establish breakpoints and then view them in the Command Window. At the `sedebg>>` prompt, enter:

```
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 0.5
Set b2 : Breakpoint for first operation at or after time 1
Set b3 : Breakpoint for first operation at or after time 1.5
```

List of Breakpoints:

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

- 3 End the debugger session. At the `sedebg>>` prompt, enter:

```
sedb.quit
```

Manipulate the structure array that is the output from `breakpoints`:

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

```
sedebg('sedemo_timeout')
```

- 2 Establish breakpoints and then record them in a structure variable. At the `sedebg>>` prompt, enter:

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

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

```
Set b1 : Breakpoint for first operation at or after time 0.5
Set b2 : Breakpoint for first operation at or after time 1
Set b3 : Breakpoint for first operation at or after time 1.5
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 End the debugger session. At the `sedbug>>` prompt, enter:

```
sedb.quit
```

More About

- “Define a Breakpoint”

See Also

`sedb.blkbreak` | `sedb.tbreak` | `sedb.evbreak` | `sedb.bdelete` | `sedb.disable`

sedb.cont

Package: sedb

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

Establish a breakpoint at $T=2$ and then run the simulation until that point:

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

```
sedebg('sedemo_timeout')
```

- 2 Establish a breakpoint and proceed in the simulation. At the `sedebg>>` prompt, enter:

```
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
```

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

```
Hit b1 : Breakpoint for first operation at or after time 2

%=====
Executing EntityGeneration Event (ev10)           Time = 2.0000000000000000
: Entity = <none>                                Priority = 1
: Block = Time-Based Entity Generator
```

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

```
sedb.quit
```

More About

- “Define a Breakpoint”
- “Use Breakpoints During Debugging”

See Also

`sedb.blkbreak` | `sedb.tbreak` | `sedb.evbreak`

sedb.currenttop

Package: sedb

Current operation in discrete-event simulation

Syntax

currenttop

Description

currenttop 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.

Examples

Compare the current operation display with the simulation log:

```
opts_struct = se_getdbopts;  
opts_struct.StartFcn={'step over','step',...  
    'step', 'currenttop','quit'};  
sedebg('sedemo_timeout',opts_struct)
```

The 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.

```

=====
Initializing Model sedemo_timeout
=====
Initializing Time-Based Entity Generators
$.....
Scheduling EntityGeneration Event (ev1)
: EventTime = 0.5000000000000000
: Priority = 1
: Entity = <none>
: Block = Time-Based Entity Generator
Independent Operation
=====
Executing EntityGeneration Event (ev1)
: Entity = <none>
: Block = Time-Based Entity Generator
Time = 0.5000000000000000
Priority = 1
$.....
Generating Entity (en1)
: Block = Time-Based Entity Generator
Dependent Operation
$.....
Entity Advancing (en1)
: From = Time-Based Entity Generator
: To = Schedule Timeout
Current Operation:
=====
Executing EntityGeneration Event (ev1)
: Entity = <none>
: Block = Time-Based Entity Generator
Time = 0.5000000000000000
Priority = 1
$.....
Entity Advancing (en1)
: From = Time-Based Entity Generator
: To = Schedule Timeout
SimEvents Debugger : Abort Simulation

```

More About

Independent Operation

An *independent operation* is one of these operations:

- 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 the application executes an event without scheduling it on the event calendar, the event 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 or a monitoring 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 an event-based signal or if the input port is not a reactive or monitoring port, the update is not an independent operation.

For more information, see “Independent Operations and Consequences in the Debugger”.

Dependent Operation

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

- “Simulation Log in the Debugger”
- “Independent Operations and Consequences in the Debugger”
- “Inspect the Current Point in the Debugger”

sedb.detail

Package: sedb

Customize debugger simulation log in discrete-event simulation

Syntax

```
detail(settingtype)
detail(struc)
detail(Name, Value)
prev = detail(settingtype)
prev = detail(struc)
prev = detail(Name, Value)
detail
curr = detail
```

Description

`detail(settingtype)` configures the debugger to omit or show certain simulation log messages. Choices of `settingtype` are described in the table.

Value of <code>settingtype</code>	Description
'none' or 'off'	Configures the debugger to omit all simulation log messages, except upon reaching breakpoints. This syntax is the same as <code>detail('en', 0, 'ev', 0, 'cal', 0)</code> . When the debugger omits all simulation log messages, you cannot use the <code>step</code> function to step to anything other than breakpoints.
'default' or 'on'	Resets the detail settings to the default values of the debugger. This syntax is the same as <code>detail('en', 1, 'ev', 1, 'sig', 1, 'cal', 0)</code> .
'all'	Configures the debugger to show all simulation log messages. This syntax is the same as <code>detail('en', 1, 'ev', 1, 'sig', 1, 'cal', 1)</code> .

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

`detail(Name, Value)` configures the debugger to show or omit certain kinds of messages in the simulation log. You can specify one, two, or three Name, Value pair arguments.

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

`detail` displays the current detail settings.

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

Input Arguments

settingtype

String that specifies a group of detail settings. Choices are 'none', 'default', and 'all'.

struc

Structure having three fields describing the detail settings that you want. Field names and field values are the same as in “Name-Value Pair Arguments” on page 1-42.

Name-Value Pair Arguments

Specify optional comma-separated pairs of Name, Value arguments. **Name** is the argument name and **Value** is the corresponding value. **Name** must appear inside single quotes (' '). You can specify several name and value pair arguments in any order as Name1, Value1, . . . , NameN, ValueN.

'cal'

A value of 1 causes the debugger to show event calendar information messages. A value of 0 causes the debugger to omit event calendar information messages, except upon reaching a breakpoint.

'en'

A value of 1 causes the debugger to show entity operation messages. A value of 0 causes the debugger to omit entity operation messages, except upon reaching a breakpoint.

When the value is 0, you cannot rely on the `step` function to step to an operation that corresponds to an omitted message.

'ev'

A value of 1 causes the debugger to show event operation messages. A value of 0 causes the debugger to omit event operation messages, except in these situations:

- The debugger reaches a breakpoint.
- The debugger is about to execute an event on the event calendar, and at least one detail setting is 1.

When the value is 0, you cannot rely on the `step` function to step to an operation that corresponds to an omitted message.

'sig'

A value of 1 causes the debugger to show signal operation messages. A value of 0 causes the debugger to omit signal operation messages, except upon reaching a breakpoint.

Output Arguments

curr

Structure that describes the current detail settings. Field names and field values are the same as in “Name-Value Pair Arguments” on page 1-42.

prev

Structure that describes the previous detail settings before changing them. Field names and field values are the same as in “Name-Value Pair Arguments” on page 1-42.

Examples

Configure displays for breakpoints and stepping:

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

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

- 2 Establish a breakpoint. Then proceed in the simulation, suppressing the simulation log until the debugger reaches the breakpoint. 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

prev =

    ev: 1
    en: 1
    sig: 1
    cal: 0

Hit b1 : Breakpoint for first operation at or after time 2

%=====
Executing EntityGeneration Event (ev10)           Time = 2.0000000000000000
: 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. This configuration makes subsequent `step` operations more informative. Then 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
Signal Operations (sig) : 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 = 3.0000000000000000
: 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
%.....%
Executing Scope
: Block = Number of Entities Time-Stamped
%.....%
Scheduling EntityGeneration Event (ev13)
: EventTime = 2.500000000000000
: Priority   = 1
: Entity     = <none>
: Block      = Time-Based Entity Generator
%=====
Executing Timeout Event (ev5)                Time = 2.000000000000000
: Entity = en2                               Priority = 1700
: Block  = Infinite Server

```

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

```
sedb.quit
```

More About

- “Customize the Debugger Simulation Log”

See Also

`sedb.blkbreak` | `sedb.cont` | `sedb.evbreak` | `sedb.tbreak`

sedb.disable

Package: sedb

Disable breakpoints in discrete-event simulation

Syntax

```
disable id1 id2 id3 ...  
disable(id_array)  
disable all
```

Description

`disable id1 id2 id3 ...` disables the breakpoint having identifier `id`. 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.

Input Arguments

id

String that represents a breakpoint identifier.

id_array

Cell array of strings that represent breakpoint identifiers.

Examples

Manipulate timed breakpoints and run the simulation until a breakpoint:

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

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

- 2 Establish and view breakpoints. At the prompt, enter:

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

The output lists the breakpoints:

```
Set b1 : Breakpoint for first operation at or after time 0.5
Set b2 : Breakpoint for first operation at or after time 1
Set b3 : Breakpoint for first operation at or after time 1.5
```

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
Disabled b3 : Breakpoint for first operation at or after time 1.5
Disabled b1 : Breakpoint for first operation at or after time 0.5
Enabled b3 : Breakpoint for first operation at or after time 1.5
```

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 1.5
```

```
%=====
Executing EntityGeneration Event (ev7)           Time = 1.5000000000000000
: Entity = <none>                                Priority = 1
: Block = Time-Based Entity Generator
```

5 End the debugger session. At the prompt, enter:

```
sedb.quit
```

More About

- “Use Breakpoints During Debugging”

See Also

`sedb.breakpoints` | `sedb.enable` | `sedb.bdelete`

sedb.enable

Package: sedb

Enable breakpoints in discrete-event simulation

Syntax

```
enable id1 id2 id3 ...  
enable(id_array)  
enable all
```

Description

`enable id1 id2 id3 ...` enables the breakpoint having identifier `id`. 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.

Input Arguments

id

String that represents a breakpoint identifier.

id_array

Cell array of strings that represent breakpoint identifiers.

Examples

Manipulate timed breakpoints and run the simulation until a breakpoint:

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

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

- 2 Establish and view breakpoints. At the prompt, enter:

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

The output lists the breakpoints:

```
Set b1 : Breakpoint for first operation at or after time 0.5  
Set b2 : Breakpoint for first operation at or after time 1  
Set b3 : Breakpoint for first operation at or after time 1.5
```

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  
Disabled b3 : Breakpoint for first operation at or after time 1.5  
Disabled b1 : Breakpoint for first operation at or after time 0.5  
Enabled b3 : Breakpoint for first operation at or after time 1.5
```

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 1.5
%=====
Executing EntityGeneration Event (ev7)           Time = 1.5000000000000000
: Entity = <none>                               Priority = 1
: Block = Time-Based Entity Generator
```

- 5 End the debugger session. At the prompt, enter:

```
sedb.quit
```

More About

- “Use Breakpoints During Debugging”

See Also

`sedb.breakpoints` | `sedb.disable` | `sedb.bdelete`

sedb.enbreak

Package: sedb

Set breakpoint on operation involving target entity

Syntax

```
enbreak(enid)
enbreak enid
benid = enbreak(enid)
```

Description

`enbreak(enid)` or `enbreak enid` sets a breakpoint for execution or cancellation of the entity with identifier `enid` in the simulation.

`benid = enbreak(enid)` returns the identifier of the entity breakpoint.

Input Arguments

enid

String that represents an entity identifier.

Default:

Output Arguments

benid

String that represents an entity breakpoint identifier.

Examples

Set a breakpoint on an advancing entity.

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

```
sedebg('sedemo_timeout')
```

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

```
step over
```

The output is:

```

=====
Initializing Time-Based Entity Generators
%.
Scheduling EntityGeneration Event (ev1)
: EventTime = 0.5000000000000000
: Priority   = 1
: Entity    = <none>
: Block     = Time-Based Entity Generator
=====
Executing EntityGeneration Event (ev1)           Time = 0.5000000000000000
: Entity = <none>                               Priority = 1
: Block  = Time-Based Entity Generator

```

- 3 Proceed further on in the simulation. At the `sedebg>>` prompt, again, enter:

```
step over
```

The output is:

```

%.
Generating Entity (en1)
: Block = Time-Based Entity Generator
%.
Entity Advancing (en1)
: From = Time-Based Entity Generator
: To   = Schedule Timeout
%.
Scheduling Timeout Event (ev2)
: EventTime = 1.5000000000000000
: Priority   = 1700
: Entity    = en1
: Block     = Schedule Timeout
%.
Entity Advancing (en1)
: From = Schedule Timeout
: To   = Infinite Server
%.
Scheduling ServiceCompletion Event (ev3)
: EventTime = 0.596864516245641
: Priority   = 1
: Entity    = en1
: Block     = Infinite Server
%.
Executing Scope
: Block = Number of Entities Time-Stamped
%.
Scheduling EntityGeneration Event (ev4)
: EventTime = 1.0000000000000000
: Priority   = 1

```

```
      : Entity      = <none>
      : Block       = Time-Based Entity Generator
%=====
Executing ServiceCompletion Event (ev3)           Time = 0.596864516245641
: Entity = en1                                   Priority = 1
: Block  = Infinite Server
```

4 Set a breakpoint that causes the debugger to stop when it is about to advance the entity, `en1`.

5 The output confirms the operation:

```
Set b1 : Breakpoint for operation involving entity en1
```

6 Run the simulation until the breakpoint:

```
cont
```

7 The output shows that the debugger hits the entity and stops upon hitting the breakpoint at entity, `en1`.

```
Hit b1 : Breakpoint for operation involving entity en1
```

```
%.....%
Entity Advancing (en1)
: From = Infinite Server
: To   = Cancel Timeout
```

8 End the debugger session. At the prompt, enter:

```
sedb.quit
```

More About

- “Define a Breakpoint”

See Also

| `sedb.cont` | `sedb.breakpoints`

sedb.eninfo

Package: sedb

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 an entity. `enid` is the identifier of the entity.

`en_struct = eninfo(enid)` returns a structure that stores information about the entity.

Input Arguments

enid

String that represents an entity identifier.

Output Arguments

en_struct

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

Field	Description
Attributes	Structure whose field names and values match the names and values of the attributes of the entity
Timers	Structure array, of which each element has these fields: <ul style="list-style-type: none"> • Tag — Timer tag • ElapsedTime — Time elapsed since the timer started
Timeouts	Structure array, of which each element has these fields: <ul style="list-style-type: none"> • Tag — Timeout tag • TimeOfTimeout — Scheduled time of timeout event • Event— Identifier of timeout event

Examples

View the values of the attributes of an entity:

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

```
sedebg('sedemo_star_routing')
```

- 2 Establish a breakpoint and proceed in the simulation. At the prompt, enter:

```
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 0.1

%=====
Executing ServiceCompletion Event (ev2)           Time = 0.3000000000000000
: Entity = en1                                   Priority = 5
: Block = Distribution Center/Infinite Server
```

- 3 Get the ID of an entity:

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

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

```
enid =
```

```
en1
```

- 4 Use the identifier, `enid`, to display information about the entity in the Command Window:

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

The output is:

```
Entity (en1) Current State          T = 0.3000000000000000
Location: Distribution Center/Infinite Server
```

```
Attributes:
Name           Value
CurrentRoute   1
CurrentServiceTime 2
CurrentStep    2
JobClass       1
JobID          1
JobServiceStatus [1x15]
LastServiceLocation 0
ServiceProcess [1x6]
ServiceTime    [1x6]
```

```
Timeouts, Timers: None
```

- 5 Store the information in variables:

```
% 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:

```
endetails =
    Time: 0.3000
    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: [1x15 double]
LastServiceLocation: 0
ServiceProcess: [1 2 4 2 3 5]
ServiceTime: [2 1 5 3 4 0]
```

```
enServiceProcess =
```

```
1 2 4 2 3 5
```

6 End the debugger session. At the prompt, enter:

```
sedb.quit
```

More About

- “Obtain Identifiers of Entities, Blocks, and Events”
- “Inspect Entities, Blocks, and Events”

See Also

`sedb.gcen` | `sedb.blkinfo` | `sedb.evinfo` | `sedb.currenttop`

sedb.evbreak

Package: sedb

Set breakpoint for execution or cancellation of event

Syntax

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

Description

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

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

Input Arguments

evvid

String that represents an event identifier.

Output Arguments

bid

String that represents a breakpoint identifier.

Examples

Set a breakpoint on a service completion event:

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

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

- 2 Proceed in the simulation. At the prompt, enter:

```
step over
```

The output is:

```

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

```

- 3 Proceed further:

```
step over
```

The output shows that a service completion event with identifier, `ev2`, appears on the event calendar. The scheduled time of the event is =1.5.

```

%.....%
Generating Entity (en1)
: Block = Time-Based Entity Generator 3
%.....%
Entity Advancing (en1)
: From = Time-Based Entity Generator 3
: To   = Set Attribute
%.....%
Setting Attribute on Entity (en1)
: myServiceTime = 1.5
: Block = Set Attribute
%.....%
Entity Advancing (en1)
: From = Set Attribute
: To   = N-Server
%.....%
Scheduling ServiceCompletion Event (ev2)
: EventTime = 1.500000000000000
: Priority   = 500
: Entity    = en1
: Block     = N-Server
%.....%
Executing Scope
: Block = Num. of Entities 2
%.....%
Scheduling EntityGeneration Event (ev3)
: EventTime = 1.000000000000000

```

```

: Priority = 500
: Entity = <none>
: Block = Time-Based Entity Generator 3
%=====
Initializing Time-Based Entity Generators
%.....%
Scheduling EntityGeneration Event (ev4)
: EventTime = 0.000000000000000 (Now)
: Priority = 500
: Entity = <none>
: Block = Time-Based Entity Generator 1
%=====
Executing EntityGeneration Event (ev4)           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, **ev2**:

```
evbreak ev2
```

The output confirms the operation:

```
Set b1 : Breakpoint for execution or cancelation of event ev2
```

- 5 Run the simulation until the breakpoint:

```
cont
```

The partial output shows that the debugger proceeds through a different event at =1, and stops upon hitting the breakpoint at event **ev2**. 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. A timed breakpoint at =1 would have caused the debugger to stop upon the first event at this time.

```

%=====
Executing EntityGeneration Event (ev9)           Time = 1.000000000000000
: Entity = <none>                               Priority = 500
: Block = Time-Based Entity Generator
%.....%
Generating Entity (en6)
: Block = Time-Based Entity Generator
%.....%
Entity Advancing (en6)
: From = Time-Based Entity Generator
: To = Single Server
%.....%
Scheduling ServiceCompletion Event (ev15)
: EventTime = 2.000000000000000
: Priority = 500
: Entity = en6
: Block = Single Server
%.....%
Scheduling EntityGeneration Event (ev16)
: EventTime = 3.000000000000000
: Priority = 500
: Entity = <none>

```

```
      : Block      = Time-Based Entity Generator

Hit b1 : Breakpoint for execution or cancelation of event ev2

%=====
Executing ServiceCompletion Event (ev2)          Time = 1.5000000000000000
: Entity = en1                                   Priority = 500
: Block = N-Server
```

- 6 End the debugger session. At the prompt, enter:

```
sedb.quit
```

More About

- “Define a Breakpoint”

See Also

`sedb.evcal` | `sedb.cont` | `sedb.breakpoints` | `sedb.bdelete`

sedb.evcal

Package: sedb

Event calendar of discrete-event simulation

Syntax

```
evcal
cal_struct = evcal
evcal enID
evcal evID
evcal blkID
evcal blkname
evcal all
evcal merged
```

Description

`evcal` displays the list of events on the event calendar of the discrete-event system in progress. If no discrete-event system is in progress, this command displays the event calendar for each discrete-event system in the model. The listing for the event in progress or the event selected for execution includes the characters `=>`. These characters appear 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 events in the event calendar of the requested discrete-event system, `cal_struct`.

`evcal enID` displays the list of events on the event calendar for the discrete-event system that contains the entity, *enID*.

`evcal evID` displays the list of events on the event calendar for the discrete-event system that contains the event, *evID*.

`evcal blkID` displays the list of events on the event calendar for the discrete-event system that contains the block identifier, *blkID*.

`evcal blkname` displays the list of events on the event calendar for the discrete-event system that contains the block name, *blkname*.

`evcal all` displays all event calendars for each discrete-event system in the model with a list of pending events in each event calendar.

`evcal merged` displays a merged list of events from all event calendars in the model. This function displays the events in the order in which they will execute.

Output Arguments

`cal_struct`

Structure that stores information about the event calendar. The `cal_struct` fields are in the following table.

Field	Description
Time	Current simulation time
ExecutingEvent	Structure that describes the event in progress or selected for execution. The structure has these fields: <ul style="list-style-type: none">• 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
DiscreteEventSystem	Discrete-event system ID of the containing discrete-event system.
PendingEvents	Structure array that describes events that are not in progress or selected for execution. Each structure in the array has these fields: <ul style="list-style-type: none">• ID — Event identifier• EventType — Type of event• EventTime — Scheduled time of event

Field	Description
	<ul style="list-style-type: none"> • Priority — Priority of event • Entity — Identifier of entity associated with the event • Block — Path name of block that executes the event

Examples

View the event calendar and then manipulate a structure variable that stores the same information:

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

```
sedebg('sedemo_timeout')
```

- 2 Establish a breakpoint and proceed in the simulation. At the `sedebg>>` prompt, enter:

```
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

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

Hit b1 : Breakpoint for first operation at or after time 2

%=====
Executing EntityGeneration Event (ev10)           Time = 2.0000000000000000
: Entity = <none>                                 Priority = 1
: Block = Time-Based Entity Generator
```

- 1 View the event calendar:

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

The following output lists all events on the event calendar for the current discrete-event system:

```
Event Calendar for Discrete-Event System ID: 0 at T = 2.000000000000000
%-----%
Discrete-Event System ID: 0 Highlight
  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
```

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

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

The output is:

```
cal_struct =
           Time: 2
  DiscreteEventSystemID: 0
      ExecutingEvent: [1x1 struct]
      PendingEvents: [2x1 struct]

ev5struct =
      ID: 'ev5'
  EventType: 'Timeout'
  EventTime: 2
      Priority: '1700'
      Entity: 'en2'
      Block: 'sedemo_timeout/Infinite Server'
```

- 3 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'
```

- 4 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
```

- 5 End the debugger session. At the `sedbug>>` prompt, enter:

```
sedb.quit
```

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

More About

- “View the Event Calendar”

See Also

`sedb.evinfo`

sedb.evinfo

Package: sedb

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`. To obtain a list of events on the event calendar and their identifiers, use `sedb.evcal`.

Input Arguments

evid

String that represents an event identifier.

Output Arguments

ev_struct

Structure that stores information about the event. The following table describes the `ev_struct` fields.

Field	Description
ID	Event identifier
EventType	Type of event
EventTime	Scheduled time of event

Field	Description
Priority	Priority of event
Entity	Identifier of entity associated with the event
Block	Path name of block that executes the event

Examples

View event information:

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

```
sedbug('sedemo_count_attributes')
```

- 2 Proceed in the simulation. At the `sedbug>>` 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
: Entity    = <none>
: Block     = Time-Based Entity Generator
```

- 3 View information about the event that the application is about to schedule:

```
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: 'sedemo_count_attributes/Time-Based Entity Generator'
```

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

```
sedb.quit
```

More About

- “Inspect Entities, Blocks, and Events”

See Also

`sedb.gcev` | `sedb.evcal`

sedb.gceb

Package: sedb

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 an empty string. If an entity is advancing, blkname indicates the block from which the entity departs.

Examples

View information about a block:

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

```
sedbug('sedemo_start_timer_read_timer')
```

- 2 Establish a breakpoint and proceed in the simulation. At the prompt, enter:

```
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 2.51
```

```
%=====
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 =
blk4
```

- 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 display information about the block and the status of entities in it:

```
% Display information in Command Window.
blkinfo(blkid)
```

The output is:

```
InfiniteServer Current State          T = 2.581301297500028
Block (blk4): 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
```

- 6 Store the information in variables:

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

The output is:

```

blkdetails =
    Time: 2.5813
    Block: 'sedemo_start_timer_read_timer/Infinite Server'
    BlockID: 'blk4'
    BlockType: 'InfiniteServer'
    Capacity: Inf
    Entities: [1x3 struct]

```

```

blkentities =
    'In Service'
    'Advancing'
    'In Service'

```

- 7** 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

```

- 8** End the debugger session. At the prompt, enter:

```

sedb.quit

```

More About

- “Inspect the Current Point in the Debugger”

See Also

`sedb.blkinf`

sedb.gcebid

Package: sedb

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 an empty string. If an entity is advancing, blkid indicates the block from which the entity departs.

Examples

View information about a block:

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

```
sedebg('sedemo_start_timer_read_timer')
```

- 2 Establish a breakpoint and proceed in the simulation. At the prompt, enter:

```
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 2.51  
  
%===== %  
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 =
blk4
```

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 display information about the block and the status of entities in it:

```
% Display information in Command Window.
blkinfo(blkid)
```

The output is:

```
InfiniteServer Current State          T = 2.581301297500028
Block (blk4): 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
```

6 Store the information in variables:

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

The output is:

```
blkdetails =  
    Time: 2.5813  
    Block: 'sedemo_start_timer_read_timer/Infinite Server'  
    BlockID: 'blk4'  
    BlockType: 'InfiniteServer'  
    Capacity: Inf  
    Entities: [1x3 struct]
```

```
blkentities =  
    'In Service'  
    'Advancing'  
    'In Service'
```

- 7 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
```

- 8 End the debugger session. At the prompt, enter:

```
sedb.quit
```

More About

- “Inspect the Current Point in the Debugger”

See Also

`sedb.blkinf`

sedb.gcen

Package: sedb

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

View the values of the attributes of an entity:

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

```
sedebg('sedemo_star_routing')
```

- 2 Establish a breakpoint and proceed in the simulation. At the prompt, enter:

```
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 0.1
```

```
%=====
Executing ServiceCompletion Event (ev2)           Time = 0.3000000000000000
: Entity = en1                                   Priority = 5
: Block = Distribution Center/Infinite Server
```

- 3 Get the ID of an entity:

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

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

```
enid =  
en1
```

- 4 Use the identifier, `enid`, to display information about the entity in the Command Window:

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

The output is:

```
Entity (en1) Current State          T = 0.3000000000000000  
Location: Distribution Center/Infinite Server  
  
Attributes:  
  Name          Value  
  CurrentRoute  1  
  CurrentServiceTime  2  
  CurrentStep   2  
  JobClass      1  
  JobID         1  
  JobServiceStatus [1x15]  
  LastServiceLocation  0  
  ServiceProcess [1x6]  
  ServiceTime    [1x6]
```

```
Timeouts, Timers: None
```

- 5 Store the information in variables:

```
% 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:

```
endetails =  
  
    Time: 0.3000  
    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: [1x15 double]
    LastServiceLocation: 0
    ServiceProcess: [1 2 4 2 3 5]
    ServiceTime: [2 1 5 3 4 0]

enServiceProcess =
    1   2   4   2   3   5
```

- 6 End the debugger session. At the prompt, enter:

```
sedb.quit
```

More About

- “Inspect the Current Point in the Debugger”

See Also

sedb.eninfo

sedb.gcev

Package: sedb

Identifier of current event

Syntax

`evid = gcev`

Description

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

Examples

View the event associated with the current operation:

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

```
sedebg('sedemo_async_stateflow')
```

- 2 Establish a breakpoint and proceed in the simulation. At the prompt, enter:

```
tbreak 1  
cont
```

The output ends with the current operation, which is the execution of the event with identifier `ev9`:

```
Hit b1 : Breakpoint for first operation at or after time 1  
%===== %  
Executing ServiceCompletion Event (ev9)           Time = 1.0000000000000000  
: Entity = en3                                   Priority = 500
```

```
: Block = Asynchronous Execution/Single Serverr
```

3 View information about the event associated with the current operation:

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

The output shows the same identifier, `ev9`, in the workspace variable, `evid`, and the ID field of the structure, `evdetails`:

```
evid =
ev2

evdetails =
    ID: 'ev2'
    EventType: 'EntityGeneration'
    EventTime: 2
    Priority: '300'
    Entity: ''
    Block: 'sedemo_async_stateflow/Asynchronous Execution/Time-Based Entity Generator'
```

4 Proceed further in the simulation:

```
step out
```

The output is:

```
%.....%
Generating Entity (en2)
: Block = Asynchronous Execution/Time-Based Entity Generator
%.....%
Entity Advancing (en2)
: From = Asynchronous Execution/Time-Based Entity Generator
: To = Asynchronous Execution/Entity Departure Function-Call Generator
%.....%
Entity Advancing (en2)
: From = Asynchronous Execution/Entity Departure Function-Call Generator
: To = Asynchronous Execution/Entity Sink
%.....%
Destroying Entity (en2)
: Block = Asynchronous Execution/Entity Sink
%.....%
Executing Signal Block
: Block = Chart
%.....%
Executing Signal Block
: Block = Scope
%.....%
Scheduling EntityGeneration Event (ev3)
: EventTime = 5.000000000000000
: Priority = 300
: Entity = <none>
: Block = Asynchronous Execution/Time-Based Entity Generator
%=====
Executing EntityGeneration Event (ev3)                Time = 5.000000000000000
```

```
: Entity = <none>                                Priority = 300
: Block  = Asynchronous Execution/Time-Based Entity Generator
```

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

```
evd_next = gcev
evdetails_next = evinfo(evd_next)
```

The output refers to the event identifier, `ev10`:

```
evd_next =
ev3

evdetails_next =
      ID: 'ev3'
      EventType: 'EntityGeneration'
      EventTime: 5
      Priority: '300'
      Entity: ''
      Block: 'sedemo_async_stateflow/Asynchronous Execution/Time-Based Entity Generator'
```

- 6 End the debugger session. At the prompt, enter:

```
sedb.quit
```

More About

- “Inspect Entities, Blocks, and Events”

See Also

`sedb.evinfo` | `sedb.evcal`

sedb.quit

Package: sedb

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 by entering the command at the incorrect command prompt.

See Also

`sedb.runtoend`

sedb.runtoend

Package: sedb

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.

More About

- “Stop the Debugger”

sedb.simtime

Package: sedb

Current time in discrete-event simulation

Syntax

```
t = simtime
```

Description

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

Examples

Set a breakpoint to advance the simulation by 2 s from the current time:

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

```
sedebg('sedemo_count_attributes')
```

- 2 Use the current time to set a breakpoint. At the `sedebg>>` prompt, enter:

```
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
```

- 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
```

```
%=====
Executing EntityGeneration Event (ev3)           Time = 2.0000000000000000
: Entity = <none>                               Priority = 1
: Block = Time-Based Entity Generator
```

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

```
sedb.quit
```

More About

- “Inspect the Current Point in the Debugger”

sedb.step

Package: sedb

Single step in discrete-event simulation

Syntax

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

Description

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

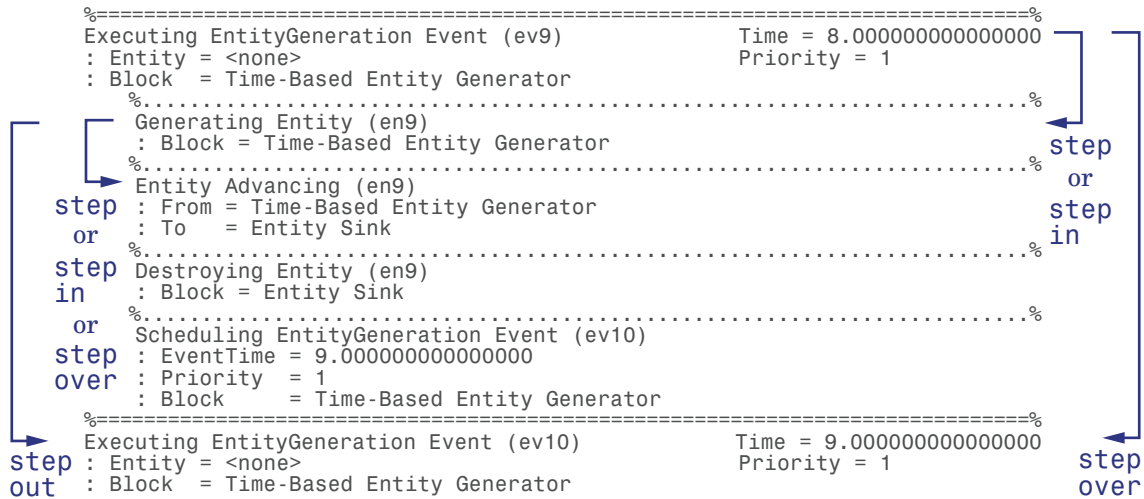
`step over` advances the simulation to the next message at the same level in the simulation log.

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

More About

Level in the Simulation Log

The figure illustrates the two-level hierarchy of the simulation log to help you distinguish among syntaxes.



Independent Operation

An *independent operation* is one of these operations:

- 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 the application executes an event without scheduling it on the event calendar, the event 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 or a monitoring 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 an event-based signal or if the input port is not a reactive or monitoring port, the update is not an independent operation.

For more information, see “Independent Operations and Consequences in the Debugger”.

- “Step Through the Simulation”

sedb.tbreak

Package: sedb

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 later enter `cont`, the simulation stops at the first operation at or after time `t`.

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

Input Arguments

t

Numeric scalar or a string that represents a number. The number is the time of a timed breakpoint.

Output Arguments

tid

String that represents a breakpoint identifier.

Examples

Manipulate timed breakpoints and run the simulation until a breakpoint:

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

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

- 2 Establish and view breakpoints. At the prompt, enter:

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

The output lists the breakpoints:

```
Set b1 : Breakpoint for first operation at or after time 0.5  
Set b2 : Breakpoint for first operation at or after time 1  
Set b3 : Breakpoint for first operation at or after time 1.5
```

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  
Disabled b3 : Breakpoint for first operation at or after time 1.5  
Disabled b1 : Breakpoint for first operation at or after time 0.5  
Enabled b3 : Breakpoint for first operation at or after time 1.5
```

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 1.5
%=====
Executing EntityGeneration Event (ev7)           Time = 1.5000000000000000
: Entity = <none>                                Priority = 1
: Block = Time-Based Entity Generator
```

5 End the debugger session. At the prompt, enter:

```
sedb.quit
```

More About

- “Define a Breakpoint”

See Also

sedb.cont | sedb.breakpoints | sedb.bdelete

sedebug

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. After you enter this command at the MATLAB command prompt, the prompt changes to `sedebug>>`. At the new prompt, you can enter debugging commands.

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

Input Arguments

sys

String that represents the top-level system of a model.

opts_struct

Structure that describes options for a SimEvents debugger session. The `StartFcn` field of the structure is a cell array of strings.

Examples

Start and end a debugger session for a SimEvents model:

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

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

The output is:

```
*** SimEvents Debugger ***  
Functions | Help | Watch Video Tutorial  
  
%===== %  
Initializing Model sedemo_count_attributes  
sedebug>>
```

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

```
sedb.quit
```

More About

- “Start the SimEvents Debugger”
- “Debugger Efficiency Tips”

See Also

`sedb.quit` | `se_getdbopts`

seupdate

Update models from previous releases

Syntax

```
seupdate('sys')  
seupdate('sys', 0)
```

Description

`seupdate('sys')` updates a SimEvents model that was created in a previous release to the current release.

This function replaces blocks in your model from previous versions of the SimEvents library with their latest version, without changing configured parameter values. If the model has associated custom libraries, the function also updates blocks in those custom libraries. The function inserts gateway blocks where a time-based signal feeds an event-based block, or where an event-based signal feeds a time-based block. If the model is not already open when you run this function, the software opens it to perform the update.

The function overwrites the model it is updating. Before beginning the update, the function completely backs up the model and associated custom libraries. The software stores this backup in a new folder, in the same directory as the model. The new folder uses the naming convention *seupdate_for_sys*.

When you run the `seupdate` function, the software provides a summary of the model and libraries it is preparing to update. The software then prompts if you are ready to proceed with the update. Enter one of the following responses:

- y
Overwrite the model and associated libraries.
- n
Cancel the operation. No changes are made to the model and associated libraries.

- help

In the SimEvents documentation, learn how the `seupdate` function changes your model.

`seupdate('sys', 0)` updates the model and referenced library components without prompting you.

Input Arguments

`sys`

Specify the name of the model to update.

Default: None

Examples

Model Migration Using `seupdate`

Use the `seupdate` function to migrate all blocks in a model to the latest version in the SimEvents library.

Run the `seupdate` function for the model that you want to migrate.

```
seupdate('ex_signalbasedevents_plots')
```

```
SEUPDATE Utility
-----
```

```
SEUPDATE runs SLUPDATE on a model and all of its associated libraries to replace
legacy blocks with blocks from SimEvents 4.0 (R2011b) or later.
```

```
Model      : ex_signalbasedevents_plots
Libraries  : --
```

```
Before you update your model:
```

1. Verify that the model compiles without any errors.
2. Learn how SEUPDATE changes your model (enter 'help' at the prompt below)

```
Update model 'ex_signalbasedevents_plots'? ([y]/n/help):|
```

Review the items under the heading `Before you update your model:` and enter a value at the prompt.

```
Update model 'ex_signalbasedevents_plots'? ([y]/n/help):y
```

The first update step checks your model file format. If either the model or any associated library files are not in the newer SLX file format (that is, they are in the older MDL format), the software prompts you to ask if you want to save the updated model in the newer format. If your model is already in the SLX format, you do not see this prompt.

```
==> NEW SLX MODEL FORMAT:
```

```
The model "ex_signalbasedevents_plots" or one or more associated library files are in MDL format.
To support new features in future releases of Simulink not supported by the MDL format,
after updating, SEUPDATE can save your MDL files in the newer SLX format.
```

```
Would you like to do this? ([y]/n/help):y
```

When you answer the file format prompt, the update process continues.

```
UPDATE : Checking file permissions ...
UPDATE : Backing up files in directory 'seupdate_for_ex_signalbasedevents_plots' ...
UPDATE : Replacing blocks with new versions ...
UPDATE : Adding gateway blocks to convert between time-based and event-based signals ...
UPDATE : Completed
```

```
Files changed : ex_signalbasedevents_plots
```

Review the report that the migration utility generates.

```
UPDATE REPORT -----
Next steps:
1. If the update did not complete, review any "ACTION NEEDED" items in this report.
2. If the update completed, learn how the update might have changed your model and libraries (read)
3. Read the more detailed update report generated by SLUPDATE (open)
END UPDATE REPORT -----
```

In this example, the model updates successfully.

```
UPDATE : Completed
Files changed : ex_signalbasedevents_plots
```

Migration of Model Containing Queue Blocks Using `seupdate`

The behavior of queue blocks changed in SimEvents version 4.0 (R2011b). If the model that you want to migrate contains legacy queue blocks, the updated queue blocks might

produce different results when you simulate the updated model. If the `seupdate` function detects that your model contains legacy queue blocks, you might see additional output in the MATLAB command window. This additional output has information on how the behavior of queue blocks has changed from previous versions of the software. The output also provides instructions to check for potential results changes from queue blocks in the updated model before you proceed with the migration.

Run the `seupdate` function for the model that you want to migrate.

```
seupdate('ex_queue_system')
```

```
>> seupdate('ex_queue_system')
```

```
SEUPDATE Utility
-----
```

```
SEUPDATE checks if your model and its associated libraries contain blocks from a
version of SimEvents prior to 4.0 (R2011b). It replaces them with the latest version
from the SimEvents library.
```

```
Model      : ex_queue_system
Libraries  : --
```

```
Before you update your model:
```

1. Verify that the model compiles without any errors.
2. Learn how SEUPDATE changes your model (enter 'help' at the prompt below)

```
This model contains SimEvents queue blocks.
```

```
In SimEvents 4.0 (R2011b), the way that the software evaluates the
'Number of entities in queue, n' parameter of queue blocks changed.
As a result of this behavior change, when you use SEUPDATE to update
your model, you are likely to see different simulation results in '#n'
output port of these blocks.
```

```
In the updated model, a results change from the '#n' output port of
a queue block occurs if the queue block is empty when an entity departs.
Use the function SEUPDATEUTIL to analyze your model for
any impact from this behavior change.
```

```
The SEUPDATEUTIL function:
```

1. Lists all queue blocks in your model with the 'Number of entities in queue, n' parameter enabled
2. Checks if the '#n' output port of the listed blocks will produce different simulation results in your updated model.

```
To perform these checks on your model, enter the following command in the
MATLAB command window:
```

```
[A] = seupdateutil(sys,'detectQueueBlockChanges')
```

```
Update model 'ex_queue_system'? ([y]/n/help):
```

If your model contains any legacy queue blocks that have the **#n** output enabled, the `seupdate` function displays the additional output that is highlighted in the graphic.

Review the report that the migration utility generates.

```
UPDATE REPORT -----
```

```
Next steps:
```

1. If the update failed, review any "ACTION NEEDED" items in this report.
2. If the update completed, learn how the update might have changed your model and libraries ([read](#))
3. Read the more detailed update report ([open](#))

```
END UPDATE REPORT -----
```

In this example, the model updates successfully.

```
Files changed : ex_queue_system
```

If you proceed with migrating your model using `seupdate`, but want to check your original model for queue blocks that produce different results in the updated version, use the backup copy of the original model that the `seupdate` function creates prior to migrating the model. The software stores the backup in a new folder, in the same directory as the model. The new folder uses the naming convention `seupdate_for_sys`.

More About

- “Check Model for Legacy Behavior”
- “Visual Appearance of Legacy SimEvents Blocks”
- “Migration Using `seupdate`”
- “Resolve Migration Failure”

simevents

Open SimEvents library

Syntax

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

Description

simevents opens the main SimEvents library.

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

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

simeventslib

Open SimEvents library

Syntax

```
simeventslib
```

Description

simeventslib opens the main SimEvents library.

Using simevents.CustomObserverInterface Objects

Monitor and visualize event and entity activity during model simulation

The `simevents.CustomObserverInterface` object monitors and visualizes events and entities during model simulation. To use this object:

- Create a new class that inherits from `simevents.CustomObserverInterface`.
- Implement the functions of interest.
- Attach the class to a model using `simevents.connectObserver`.
- Run the simulation.
- Disconnect the class from the model when done using `simevents.disconnectObserver`.

Examples

Inherit Observer Class

Inherit observer class `RestaurantAnimator` from `simevents.CustomObserverInterface`.

```
classdef RestaurantAnimator < simevents.CustomObserverInterface
properties
    mModel;
end
end
```

Object Functions

```
entityAdvance
entityCombineentityDestroyentityGenerateentityQueueentityReplicate
entitySpliteventCancelEventExecute
eventSchedulegetBlocksToObserveinitializeterminate
pauseFcncontinueFcngetPace
```

Create Object

Create `CustomObserverInterface` objects using the `simevents.CustomObserverInterface` function.

See Also

`simevents.connectObserver` | `simevents.disconnectObserver`

More About

- “Interface for Custom Visualization”

Introduced in R2015a

continueFcn

Continue observer when model simulation continues after pause

Syntax

```
continueFcn(observer,model)
```

Description

`continueFcn(observer,model)` continues the specified observer when model simulation continues after pausing. In the context of the function, a model pauses when:

- A click on the **Pause** button in the Simulink Editor
- Completion of a single step from single stepping the simulation using **Step Forward** and **Step Back** in the Simulink Editor

Examples

Continue Observer

Continue observer and print `Continuing`. `this` is the observer object. This example builds on the figure that the example in the `initialize` example sets up.

```
function continueFcn(this, modelName)
    set(this.txtHandle, 'Text', 'Continuing');
end
```

Input Arguments

observer — Observer object

string

Observer object to continue after pause, specified as a string.

model — Name of model

string

Name of model to continue after pause, specified as a string.

See Also

pauseFcn | simevents.CustomObserverInterface

Introduced in R2015a

simevents.connectObserver

Connect observer to model

Syntax

```
simevents.connectObserver(model,application)
```

Description

`simevents.connectObserver(model,application)` connects the observer defined in the specified application to the model.

Examples

Connect observer to model

Connect the observer to the model `sedemo_custom_visualization`.

```
animator = RestaurantAnimator;  
mdl = 'mRestaurant';  
open_system(mdl);
```

```
simevents.connectObserver(mdl, animator);
```

- “Custom Visualization Examples”

Input Arguments

model — Name of model

string

Name of model to connect observer to, specified as a string.

application — Observer object or variable holder observer object

string

Observer object or variable holder observer object that contains the implementation of observer functions, specified as a string.

See Also

`simevents.CustomObserverInterface` | `simevents.disconnectObserver`

Introduced in R2015a

simevents.CustomObserverInterface

Create custom observer object

Syntax

```
classdef MyClass < simevents.CustomObserverInterface
```

Description

`classdef MyClass < simevents.CustomObserverInterface` directly inherits from the `simevents.CustomObserverInterface` class to create a custom observer class, where `objectname` is the name of your object.

After creating this class, you can construct the observer object with:

```
object = MyClass
```

Then, use related functions to connect this object to a model to monitor entities and events. For example, you can develop an application to visualize entity activities and events in a model.

Examples

Create Observer

Create an observer object named `RestaurantAnimator`.

```
classdef RestaurantAnimator < simevents.CustomObserverInterface
    properties
        mModel;
    end
    .
    .
    .
function observer = RestaurantAnimator
```

```
observer.mModel = 'mRestaurant';
```

- “Custom Visualization Examples”

Input Arguments

MyClass – Observer object

string

Observer object to create, specified as a string.

See Also

simevents.CustomObserverInterface

Introduced in R2015a

entityAdvance

Detect advance of entities from one block to another

Syntax

```
entityAdvance(observer,entity,from,to)
```

Description

`entityAdvance(observer,entity,from,to)` detects advancement of entity (`entity`) from one block (`from`) to another (`to`).

The entity has not yet arrived at the destination block when the software calls this function. When implementing this function, using the `sedb.blkinfo` function on the destination block does not show this entity.

Examples

Detect Advance of Entity

Detect advance of entity from the `from` block to the `to` block. `this` is the observer object.

```
function entityAdvance(this, entity, from, to)
    fprintf('At time %.2f, entity %s is advancing from %s to %s\n', ...
        sedb.simtime, entity, from, to);
    enStruct = sedb.eninfo(entity);
    fprintf('This entity's attribute Priority = %d\n', ...
        enStruct.Attributes.Priority);
end
```

- “Custom Visualization Examples”

Input Arguments

observer — Observer object

string

Observer object, specified as a string.

entity — Unique ID of advancing entity

double

Unique ID of advancing entity, specified as a double.

from — Full path name of block

string

Full path name of block from which entity departs, specified as a string.

to — Full path name of end block

string

Full path name of end block where the entity arrives, specified as a string.

See Also

`sedb.blkinfo` | `simevents.CustomObserverInterface`

Introduced in R2015a

entityCombine

Detect combination of component entities into composite entity

Syntax

```
entityCombine(observer,componentEntities,compositeEntity,block)
```

Description

`entityCombine(observer,componentEntities,compositeEntity,block)` detects whether the component entities are combined into the specified component entity in the specified block.

Examples

Detect the Combining Entities

Detect the combining entities and print the combining equation. `this` is the observer object.

```
function entityCombine(this, componentEntities, compositeEntity, block)
    lhs = '';
    for idx = 1 : length(componentEntities)
        lhs = [lhs componentEntities{idx} ' + '];
    end
    lhs = lhs(1:end-2);
    eqn = [lhs '=> ' compositeEntity];
    fprintf('%s\n', eqn);
end
```

- “Custom Visualization Examples”

Input Arguments

observer — Observer object

string

Observer object, specified as a string.

componentEntities — IDs of entities being combined

cell array

IDs of entities being combined, specified as a cell array.

compositeEntity — Unique ID of new composite entity

double

Unique ID of new composite entity containing combined entities, specified as a double.

block — Full path name of Entity Combiner block

string

Full path name of Entity Combiner block, specified as a string.

See Also

`sedb.blkinfo` | `sedb.eninfo` | `simevents.CustomObserverInterface`

Introduced in R2015a

entityDestroy

Detect destruction of entity

Syntax

```
entityDestroy(observer,entity,block)
```

Description

`entityDestroy(observer,entity,block)` detects the destruction of the entity in the specified block.

The entity is not yet destroyed in the block when the software calls this function. When implementing this function, you can still use the `sedb.entityInfo` function to inspect the entity.

Examples

Detect the Destruction of Entity

Detect the destruction of `entity` in `block`. `this` is the observer object.

```
function entityDestroy(this, entity, block)
    % Remove entity from local cache
    idx = strcmp(this.allEntities, entity);
    this.allEntities = this.allEntities(~idx);
end
```

- “Custom Visualization Examples”

Input Arguments

observer — Observer object

string

Observer object, specified as a string.

entity — Unique ID of entity

double

Unique ID of entity being destroyed, specified as a double.

block — Full path name of block

string

Full path name of block destroying the entity, specified as a string. This argument can be empty if the output port is not connected.

See Also

`sedb.eninfo` | `simevents.CustomObserverInterface`

Introduced in R2015a

entityGenerate

Detect entity generation

Syntax

```
entityGenerate(observer,entity,block)
```

Description

`entityGenerate(observer,entity,block)` detects entity generation in the specified block.

When implementing this function, you can use the `sedb.eninfo` function to inspect the entity state. However, at the time of entity generation, the entity is unlikely to have attributes. The entity does not have an attribute until you use the Set Attribute block to set one.

Examples

Detect Generation of Entity

Detect generation of entity in block `myModel/myBlock`. `this` is the observer object.

```
function entityGenerate(this, entity, block)
    if strcmp(block, 'myModel/myBlock')
        % Add entity to a local cache
        this.allEntities = [this.allEntities, entity];
    end
end
```

- “Custom Visualization Examples”

Input Arguments

observer — Observer object

string

Observer object.

entity — Unique ID of generated entity

double

Unique ID of generated entity.

block — Full path name of block

string

Full path name of block that generates entity.

See Also

sedb.eninfo | simevents.CustomObserverInterface

Introduced in R2015a

entityQueue

Detect entity enqueueing

Syntax

```
entityQueue(observer,entity,block,queuePosition,queueSize)
```

Description

`entityQueue(observer,entity,block,queuePosition,queueSize)` detects entity enqueueing in a queue block.

The entity is not yet enqueued in the queue block when the software calls this function. When implementing this function, using the `sedb.blkinfo` function on the queue block does not show this entity.

Examples

Detect Entity Being Enqueued

Detect entity being enqueued in the `block` block. `this` is the observer object.

```
function entityQueue(this, entity, block, queuePosition, queueSize)
    fprintf('Queuing entity %s at position %d of %d in block %s\n', ...
        entity, queuePosition, queueSize, block);
end
```

- “Custom Visualization Examples”

Input Arguments

observer — Observer object

string

Observer object, specified as a string.

entity — Unique ID of enqueueing entity

double

Unique ID of enqueueing entity, specified as a double.

block — Full path name of queue block

string

Full path name of queue block, specified as a string.

queuePosition — Position of entity

double

Position of entity to be queued at, specified as a double.

queueSize — Number of entities

double

Number of entities currently in queue, specified as a double.

See Also

`sedb.blkinfo` | `simevents.CustomObserverInterface`

Introduced in R2015a

entityReplicate

Detect replication of entity

Syntax

```
entityReplicate(observer,entity,replica,block,currentOutputPort)
```

Description

`entityReplicate(observer,entity,replica,block,currentOutputPort)` detects replication of an entity, one replica at a time.

Examples

Detect Replication of Entity

Detect replication of entity as replica in block. this is the observer object.

```
function entityReplicate(this, entity, replica, block, currOutputPort)
    fprintf(['Replicating entity %s to produce new entity %s ' ...
            'in block %s (replica %d)\n'], ...
            entity, replica, block, currOutputPort);
end
```

- “Custom Visualization Examples”

Input Arguments

observer — Observer object

string

Observer object, specified as a string.

entity — Unique ID of entity being replicated

double

Unique ID of being replicated, specified as a double.

replica — Unique ID of current entity replica

double

Unique ID of current entity replica, specified as a double.

block — Full path name of Replicate block

string

Full path name of Replicate block, specified as a string.

currentOutputPort — Index of output port

double

Index of output port from which replica is departing, specified as a double.

See Also

`sedb.blkinfo` | `sedb.eninfo` | `simevents.CustomObserverInterface`

Introduced in R2015a

entitySplit

Detect composite entity splitting into component entities

Syntax

```
entitySplit(observer, compositeEntity, componentEntities, block)
```

Description

`entitySplit(observer, compositeEntity, componentEntities, block)` detects composite entity splitting into component entities in the Entity Splitter block.

Examples

Detect Entity Split

Detect entity split and print the splitting equation. `this` is the observer object.

```
function entitySplit(this, compositeEntity, componentEntities, block)
    rhs = '';
    for idx = 1 : length(componentEntities)
        rhs = [rhs componentEntities{idx} ' + '];
    end
    rhs = rhs(1:end-2);
    eqn = [compositeEntity ' => ' rhs];
    fprintf('%s\n', eqn);
end
```

- “Custom Visualization Examples”

Input Arguments

observer — Observer object

string

Observer object, specified as a string.

compositeEntity — Unique ID of composite entity

double

Unique ID of composite entity to split, specified as a double.

componentEntities — IDs of component entities

cell array

IDs of components entities to release as a result of the split from `compositeEntity`, specified as a cell array.

block — Full path name of Entity Splitter block

string

Full path name of Entity Splitter block, specified as a string.

See Also

Entity Splitter | `sedb.blkinfo` | `sedb.eninfo` | `simevents.CustomObserverInterface`

Introduced in R2015a

eventCancel

Detect when block cancels event

Syntax

```
eventCancel(observer,eventType,eventID,eventPriority,entity,block)
```

Description

`eventCancel(observer,eventType,eventID,eventPriority,entity,block)` detects when the specified block cancels the specified event.

Use the `sedb.evinfo` function to query event properties.

The event is not yet canceled from the event calendar when the software calls this function. When implementing this function, using the `sedb.evcal` function still shows this entity.

Examples

Detect When Block Cancels eventId

Detect when block cancels `eventId` for `entity`. `this` is the observer object.

```
function eventCancel(this, eventType, eventId, eventPriority, entity, block)
    if strcmp(get_param(block, 'BlockType'), 'SingleServer') && ...
        strcmp(eventType, 'ServiceCompletion')
        % Perform some processing here
    end
end
```

- “Custom Visualization Examples”

Input Arguments

observer — Observer object

string

Observer object to list blocks for, specified as a string.

eventType — Type of event

string

Type of event being executed, specified as a string. For a list of commonly used event types, see “Interface for Custom Visualization”.

eventID — ID of event

double

ID of event to cancel, specified as a double.

eventPriority — Numerical priority of event

double

Numerical priority of event being canceled, specified as a double.

entity — Unique ID of entity

double

Unique ID of entity whose event to cancel, specified as a double. If the event does not have a target entity, you can use an empty value.

block — Full path name of block

string

Full path name of block that cancels the event, specified as a string.

See Also

sedb.blkinfo | sedb.eninfo | sedb.evcal | sedb.evinfo |
simevents.CustomObserverInterface

Introduced in R2015a

eventExecute

Detect when block executes event

Syntax

```
eventExecute(observer, eventType, eventId, eventPriority, entity, block)
```

Description

`eventExecute(observer, eventType, eventId, eventPriority, entity, block)` detects when the specified block executes the specified event.

Use the `sedb.evinfo` function to query event properties.

The event has not been executed when the software calls this function. When implementing this function, using the `sedb.evcal` function to query the event calendar still shows this event as the currently executing event.

Examples

Detect When Block Executes eventId

Detect when block executes `eventId` for `entity`. `this` is the observer object.

```
function eventExecute(this, eventType, eventId, eventPriority, entity, block)
    if strcmp(get_param(block, 'BlockType'), 'SingleServer') && ...
        strcmp(eventType, 'ServiceCompletion')
            % Perform some processing here
    end
end
```

- “Custom Visualization Examples”

Input Arguments

observer — Observer object

string

Observer object, specified as a string.

eventType — Type of event

string

Type of event being executed, specified as a string. For a list of commonly used event types, see “Interface for Custom Visualization”.

eventID — Unique ID of event

double

Unique ID of event being executed, specified as a double.

eventPriority — Numerical priority of event

double

Numerical priority of event being executed, specified as a double.

entity — Unique ID of entity

double

Unique ID of entity for which event is being executed, specified as a double. If the event does not have a target entity, you can use an empty value.

block — Full path name of block

string

Full path name of block that executes the event, specified as a string.

See Also

sedb.blkinfo | sedb.eninfo | sedb.evcal | sedb.evinfo |
simevents.CustomObserverInterface

Introduced in R2015a

eventSchedule

Detect when block schedules event

Syntax

```
eventSchedule(observer,eventType,eventID,eventPriority,entity,block)
```

Description

`eventSchedule(observer,eventType,eventID,eventPriority,entity,block)` detects when the specified block schedules the specified event for the specified entity in the specified block.

Use the `sedb.evinfo` function to query event properties.

The event is not yet scheduled in the event calendar when the software calls this function. When implementing this function, using the `sedb.blkinfo` function on the queue block does not show this entity.

Examples

Detect Block Schedules an Event

Detect when block schedules an event for `entity`. `this` is the observer object.

```
function eventSchedule(this, eventType, eventId, eventPriority, entity, block)
    if strcmp(get_param(block, 'BlockType'), 'SingleServer') && ...
        strcmp(eventType, 'ServiceCompletion')
            % Perform some processing here
    end
end
```

Input Arguments

observer — Observer object

string

Observer object, specified as a string.

eventType — Type of event

string

Type of event to schedule on the event calendar, specified as a string. For a list of commonly used event types, see “Interface for Custom Visualization”.

eventID — Unique ID of event

double

Unique ID of event being scheduled, specified as a double.

eventPriority — Numerical priority of event

double

Numerical priority of event being scheduled, specified as a double.

entity — Unique ID of entity

double

Unique ID of entity for which event is being scheduled, specified as a double. If the event does not have a target entity, you can use an empty value.

block — Block

string

Block that contains the event, specified as a string.

See Also

`sedb.blkinfo` | `sedb.eninfo` | `sedb.evcal` | `sedb.evinfo` | `simevents.CustomObserverInterface`

Introduced in R2015a

getBlocksToObserve

Blocks to observe during model simulation

Syntax

```
blocks = getBlocksToObserve(observer)
```

Description

`blocks = getBlocksToObserve(observer)` lists the blocks to observe during model simulation. Use this function to receive notifications only for events associated with this list of blocks. To observe all blocks in the model, specify an empty cell array.

Examples

Get All Blocks to Observe

Observe all blocks in the model.

```
function blks = getBlocksToObserve(this)
    blks = {};
end
```

- “Custom Visualization Examples”

Input Arguments

observer — Observer object

string

Observer object to list blocks for, specified as a string.

Output Arguments

blocks — List of blocks

array

List of blocks, specified as an array.

See Also

`simevents.CustomObserverInterface`

Introduced in R2015a

getPace

Pace at which simulation runs

Syntax

```
pace = getPace(observer)
```

Description

`pace = getPace(observer)` runs the simulation at the specified pace so that the connected animations appear as continuous movement. The function calculates the pace as simulation seconds/clock seconds.

For example:

- A pace of 1 causes a simulation with a stop time of 100 to take at least 100 clock seconds, if possible.
- A pace of 2 causes a simulation with a stop time of 100 to take at least 50 clock seconds, if possible.
- A pace of 0.5 causes a simulation with a stop time of 100 to take at least 200 clock seconds, if possible.

Note:

- The SimEvents software does not honor this pace setting for large models that take more than one clock second to simulate one simulation time unit.
- This setting does not run the model in real time.

Examples

Set No Pacing

Set no pacing by default.

```
function p = getPace(this)
```

```
p = 0;  
end
```

- “Custom Visualization Examples”

Input Arguments

observer — Observer object

string

Observer object to set simulation pace for, specified as a string.

Output Arguments

pace — Pace

double

Pace to generate simulation.

See Also

`simevents.CustomObserverInterface`

Introduced in R2015a

initialize

Initialize visualization during model initialization

Syntax

```
initialize(observer,model)
```

Description

`initialize(observer,model)` initializes the visualization of the simulation at the start of model simulation. For example, you can implement this function to:

- Set up a figure window in which to display animation
- Initialize a third-party animation to receive data

Examples

Set Up Figure Window

Set up a figure window on which to display the animation and print `Initialized`. `this` is the observer object.

```
function initialize(this, modelName)
    this.model = modelName;
    this.figHandle = figure;
    this.axHandle = axes;
    set(this.axHandle, 'XLim', [0 1], 'YLim', [0 1]);
    set(this.axHandle, 'Box', 'on');
    this.txtHandle = text(0.5, 0.5, 'Initialized');
    this.allEntities = {};
end
```

Input Arguments

observer — Observer object

string

Observer object.

model — Name of model

string

Name of model whose entities and events to observe, specified as a string.

See Also

`simevents.CustomObserverInterface | terminate`

Introduced in R2015a

simevents.disconnectObserver

Disconnect observer from model

Syntax

```
simevents.disconnectObserver(model)
```

Description

`simevents.disconnectObserver(model)` disconnects the observer from the model.

Examples

Disconnect Observer from Model

Disconnect observer from the model `sedemo_custom_visualization`.

```
modelName = 'sedemo_custom_visualization';  
open_system(modelname);  
animator = RestaurantAnimator;  
simevents.connectObserver(modelname, animator);  
sim(modelname);  
waitfor(animator.getFigureHandle, 'Tag', 'End');  
simevents.disconnectObserver(modelname);
```

- “Custom Visualization Examples”

Input Arguments

model — Name of model

string

Name of model to disconnect from, specified as a string.

See Also

`simevents.connectObserver` | `simevents.CustomObserverInterface`

Introduced in R2015a

pauseFcn

Pause observer when model pauses simulation

Syntax

```
pauseFcn(observer,model)
```

Description

`pauseFcn(observer,model)` pauses the observer when the model pauses simulation. In the context of the function, a model is paused:

- With a click of the **Pause** button in the Simulink Editor
- When a single step completes from single stepping the simulation using **Step Forward** and **Step Back** in the Simulink Editor

You can implement this function to inspect components such as these when the model is paused:

- Entities contained in storage blocks
- Entity attributes
- Current simulation time

Examples

Pause Observer

Pause observer and print **Paused**. `this` is the observer object. This example builds on the figure that the example in the `initialize` example sets up.

```
function pauseFcn(this, modelName)
    set(this.txtHandle, 'Text', 'Paused');
```

end

Input Arguments

observer — Observer object

string

Observer object, specified as a string.

model — Name of model

string

Name of paused model, specified as a string.

See Also

`continueFcn` | `sedb.blkinfo` | `sedb.eninfo` | `sedb.simtime` |
`simevents.CustomObserverInterface`

Introduced in R2015a

terminate

Terminate animation when model terminates

Syntax

```
terminate(observer,model)
```

Description

`terminate(observer,model)` terminates animation activities when the model simulation stops. For example, you can implement this function to:

- Inspect any remaining entities in storage blocks
- Run postprocessing on collected data
- Stop MATLAB timer objects controlling animation
- Autoscale figure windows displaying results

Examples

Terminate the Animation

Terminate the animation and print `Terminated`. `this` is the observer object. This example builds on the figure that the example in the `initialize` example sets up.

```
function terminate(this, modelName)
    set(this.txtHandle, 'Text', 'Terminated');
end
```

Input Arguments

observer — Observer object

string

Observer object, specified as a string.

model — Name of model

string

Name of model that the observer is monitoring, specified as a string.

See Also

sedb.blkinfo | sedb.eninfo | simevents.CustomObserverInterface

Introduced in R2015a

Blocks — Alphabetical List

Attribute Function

Access and modify attributes using MATLAB code

Library

Attributes



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

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

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

Timing and Connections

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

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

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Entity Output Ports

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

Examples

- “Set Attributes”
- “Incorporate Legacy Code”
- “Unique and Accessible Attribute Names in Composite Entity”
- Modeling Multicomponent Parts using Combiners and Splitters example
- Distributing Multi-Class Jobs to Service Stations example, within the Distribution Center subsystem and its Service History Monitor subsystem

See Also

Set Attribute, Get Attribute

“Write Functions to Manipulate Attributes”

Attribute Scope

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.

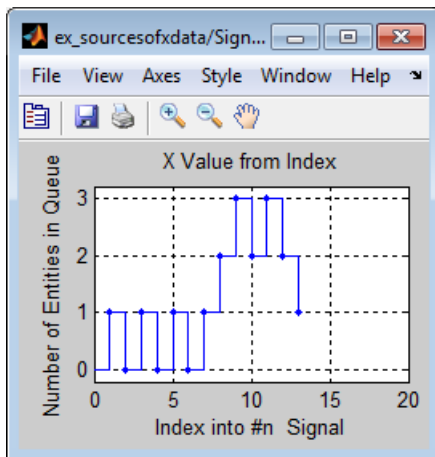
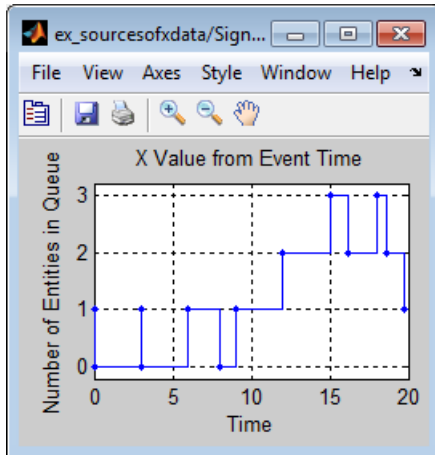
The **Plot type** parameter on the **Plotting** tab determines whether and how the block connects the points that it plots. For details, see “Connections Among Points in Plots”.

Selecting Data for the Horizontal Axis

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

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

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



Ports

Entity Input Ports

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

Entity Output Ports

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

Signal Output Ports

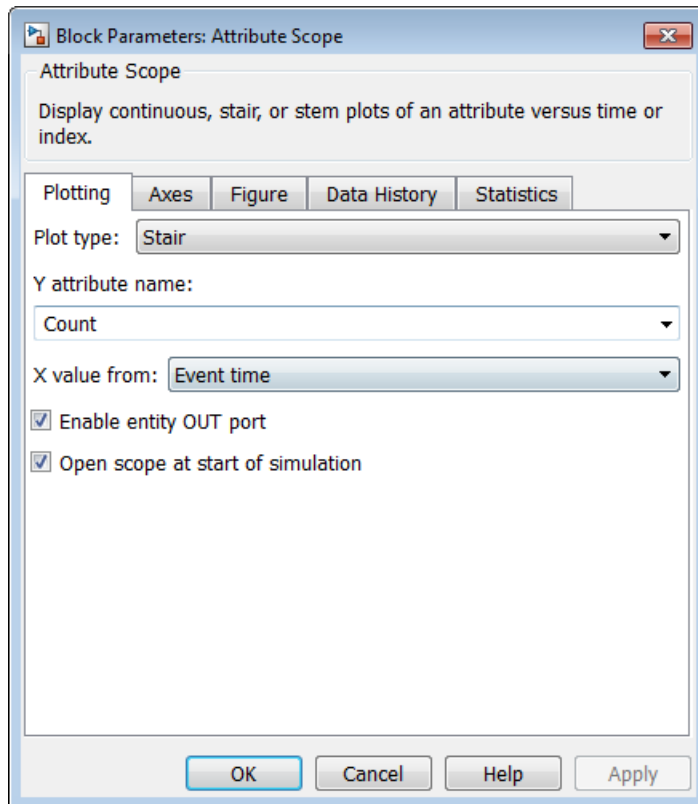
Label	Description
#a	Number of entities that have arrived at the block since the start of the simulation.

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

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” 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 2-4 for details.

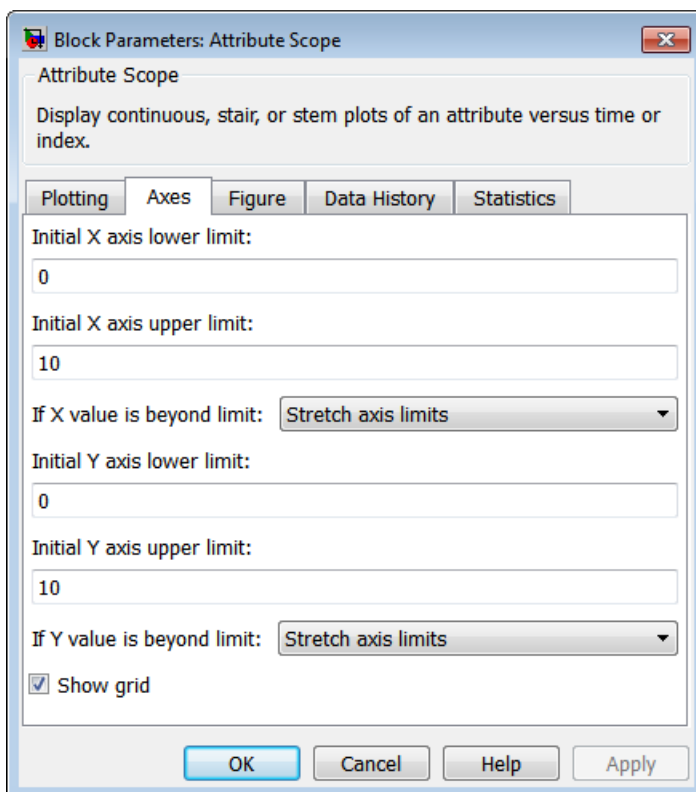
Enable entity OUT port

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

Open scope at start of simulation

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

Axes 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 “Vary Axis Limits Automatically”.

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.

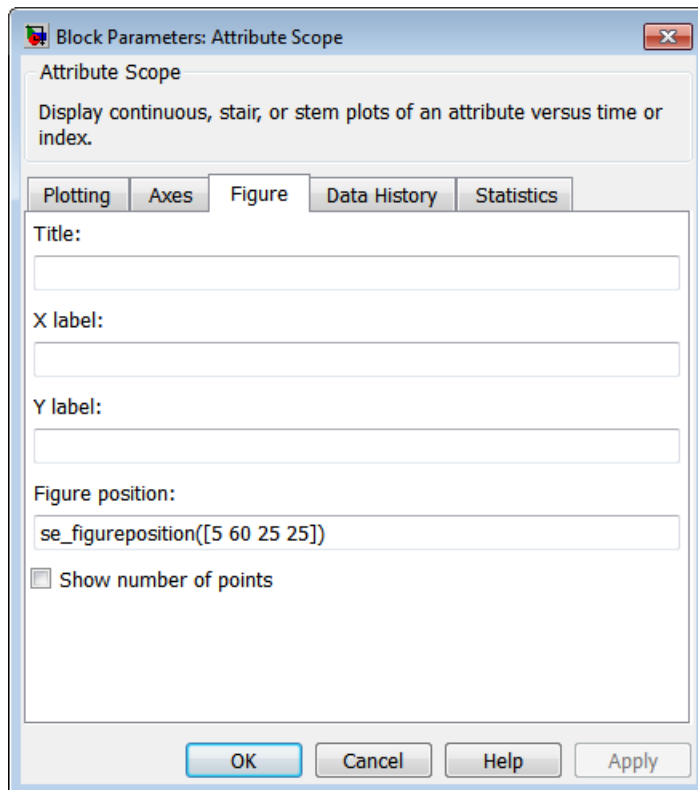
If Y value is beyond limit

Determines how the plot changes if one or more attribute values are not within the limits shown on the Y axis. For details, see “Vary Axis Limits Automatically”.

Show grid

Toggles the grid on and off.

Figure Tab



Title

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

Y label

Text that appears to the left of the vertical axis.

X label

Text that appears below the horizontal axis.

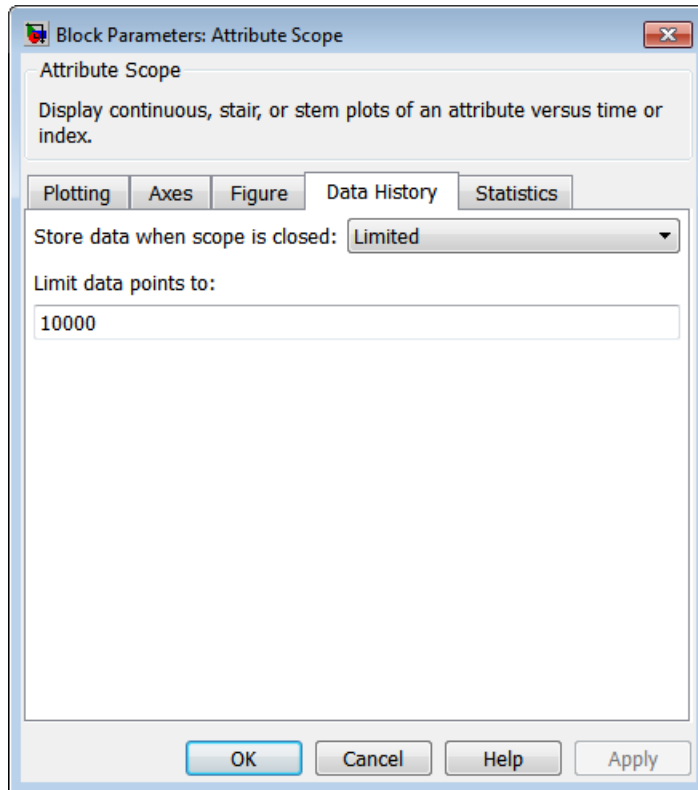
Position

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

Show number of entities

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

Data History Tab



Store data when scope is closed

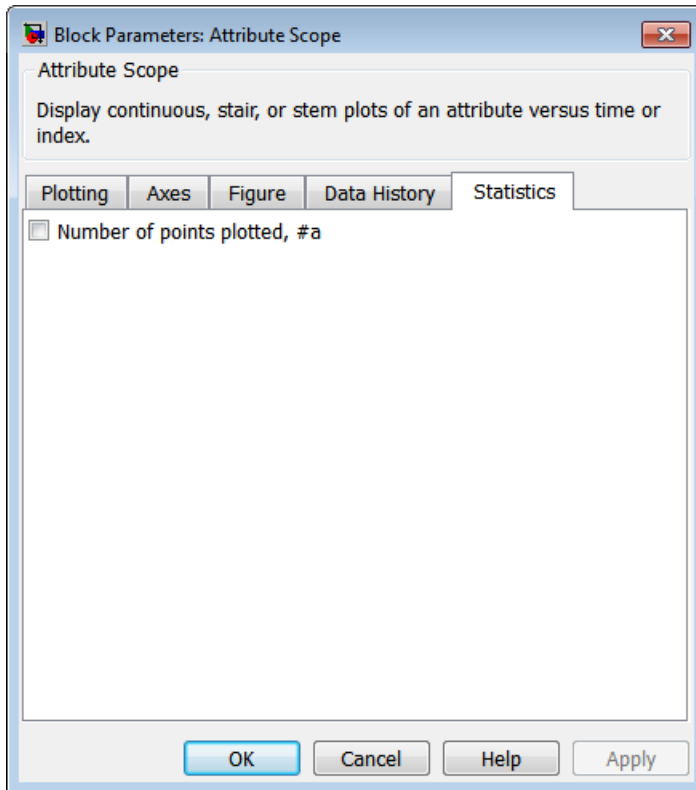
Select **Unlimited** to have the block cache all data for future viewing, **Limited** to cache a portion of the most recent data, and **Disabled** to avoid caching undisplayed data.

Limit data points to

The number of data points the block caches, using the most recent data. You see this field only if you set **Store data when scope is closed** to **Limited**.

Statistics Tab

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



Number of entities arrived

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

Examples

- “Round-Robin Approach to Choosing Inputs”
- “Set Attributes”

See Also

X-Y Attribute Scope, Signal Scope

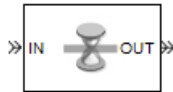
“Choose and Configure Plotting Blocks”, “Access Entity Attributes”

Cancel Timeout

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

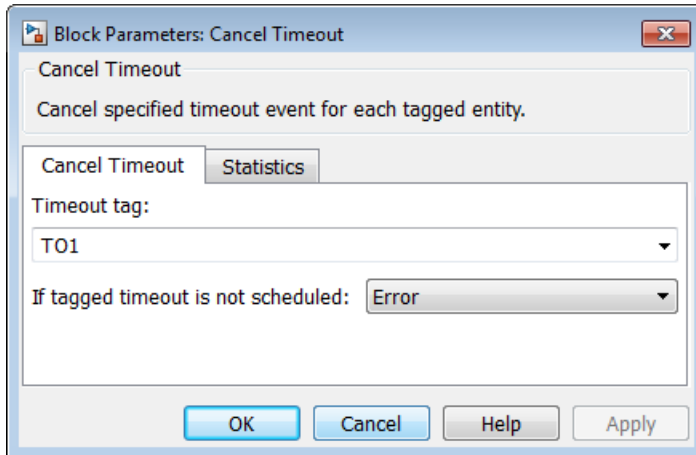
Label	Description	Time of Update When Statistic is On	Order of Update
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	3
#t	Number of entities that have departed from this block and been associated with a timeout of the specified name.	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

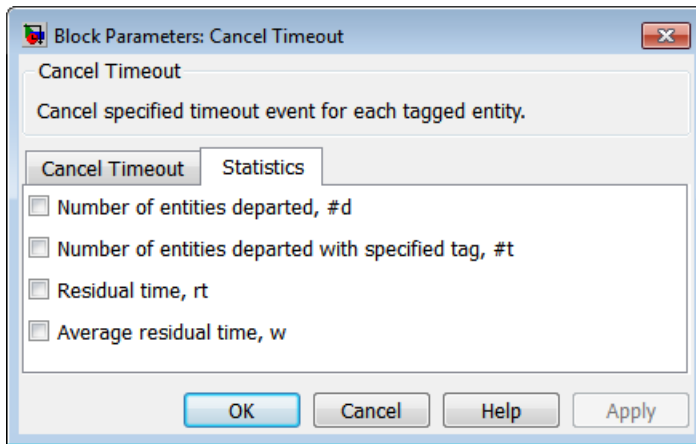
Name of the timeout event to cancel, corresponding to the **Timeout tag** parameter of a Schedule Timeout block in the model.

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 the block produces data at signal output ports or omits those ports.



Number of entities departed

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

Number of entities departed with specified tag

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

Residual time

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

Average residual time

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

Examples

- “Use Timeouts to Limit Entity Queueing Time”
- “Define Entity Timeout Paths”
- “Reroute Timed-Out Entities to Expedite Handling”
- “Limit the Time Until Service Completion”

See Also

Schedule Timeout

“Workflow for Using Timeouts”

Conn

Provide entity input port or entity output port for virtual subsystem

Library

SimEvents Ports and Subsystems

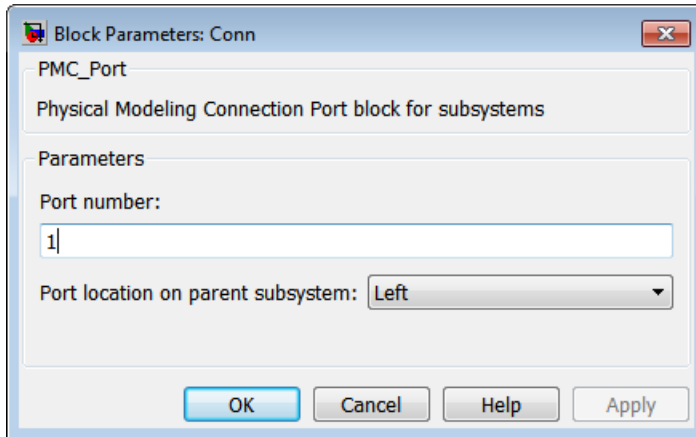
Description

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

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

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

Dialog Box



Port number

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

Port location on parent subsystem

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

Discrete Event Signal to Workspace

Write event-based signal to workspace

Library

SimEvents Sinks



Description

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

This block is similar to the To Workspace block in the Simulink Sinks library but is tailored for use with event-based signals.

Output Format

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

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

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

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

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

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

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

Comparison with To Workspace Block

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

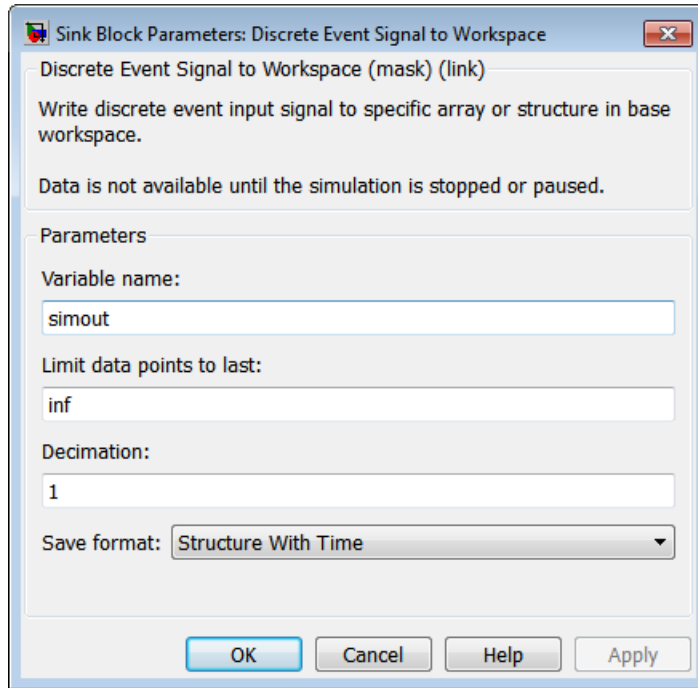
The simulation times at which this block records data is typically unrelated to the variable that a model creates if you select **Time** in the **Save to workspace** section of the **Data Import/Export** tab of the Configuration Parameters dialog box. By default, this option is selected and the variable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

Ports

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

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

Dialog Box



Variable name

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

Limit data points to last

The maximum number of input samples to be saved.

Decimation

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

Save format

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

Examples

- “Send Queue Length to the Workspace”
- “Observe Service Completions”

See Also

To Workspace

“Send Data to the MATLAB Workspace”

Enabled Gate

Allow entity arrivals upon positive control signal

Library

Gates

Description



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

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Signal Input Ports

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

Entity Output Ports

Label	Description
OUT	Port for departing entities.

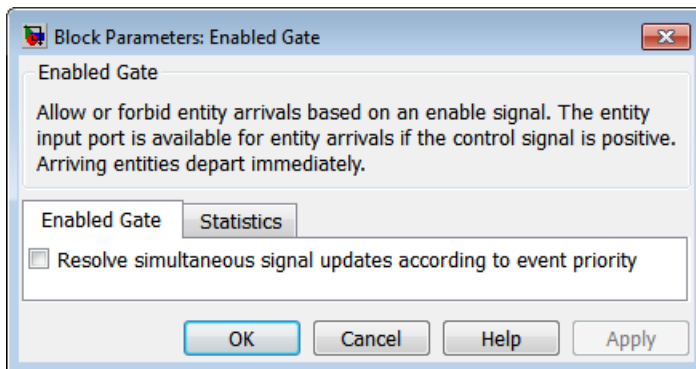
Signal Output Ports

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

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

Dialog Box

Enabled Gate Tab



Resolve simultaneous signal updates according to event priority

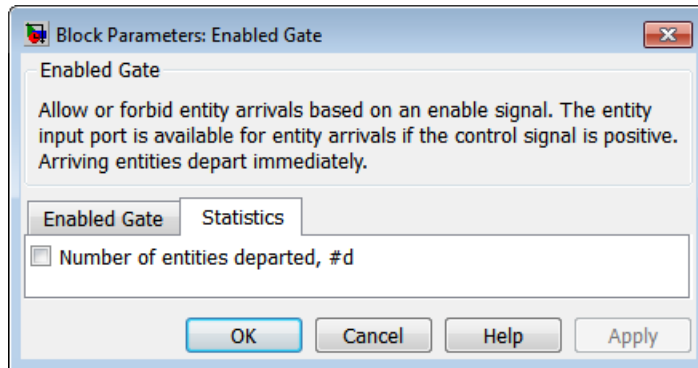
Select this option to prioritize the gate-opening or gate-closing event explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority SYS1 on the event calendar. For details, see “Resolve Simultaneous Signal Updates”.

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 “Specify Event Priorities to Resolve Simultaneous Signal Updates”. You see this field only if you select **Resolve simultaneous signal updates according to event priority**.

Statistics Tab

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



Number of entities departed

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

Examples

- “Control Joint Availability of Two Servers”
- “Treat First Entity as Special Case”
- “Limit the Time Until Service Completion”

See Also

Release Gate

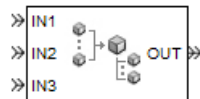
“Role of Gates in SimEvents Models”

Entity Combiner

Generate one entity per set of entities arriving simultaneously

Library

Entity Management



Description

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

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

Waiting for Component Entities on Multiple Paths

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

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

Managing Information When Combining Entities

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

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

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

Options for Managing Information When Combining Entities

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

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

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

Ports

Entity Input Ports

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

Entity Output Ports

Label	Description
OUT	Port for departing entities.

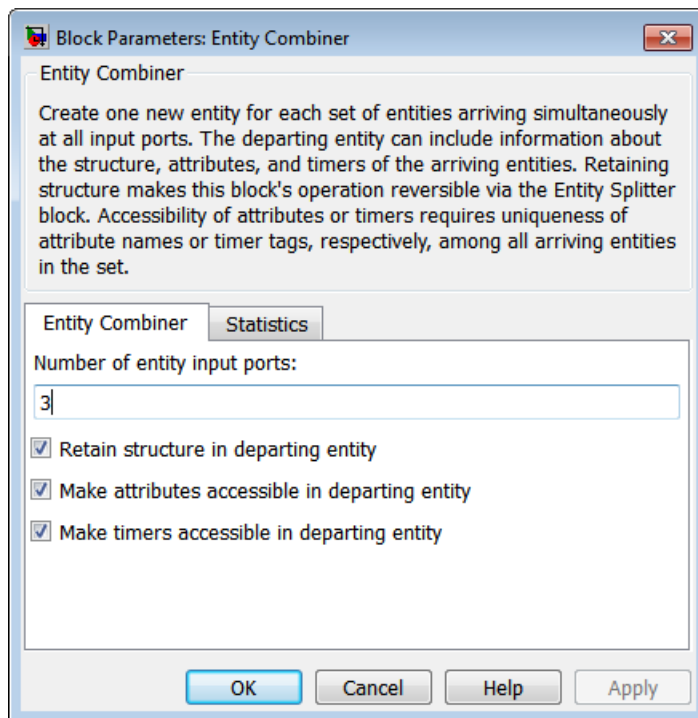
Signal Output Ports

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

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

Dialog Box

Entity Combiner Tab



Number of entity input ports

Determines how many entity input ports the block has.

Retain structure in departing entity

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

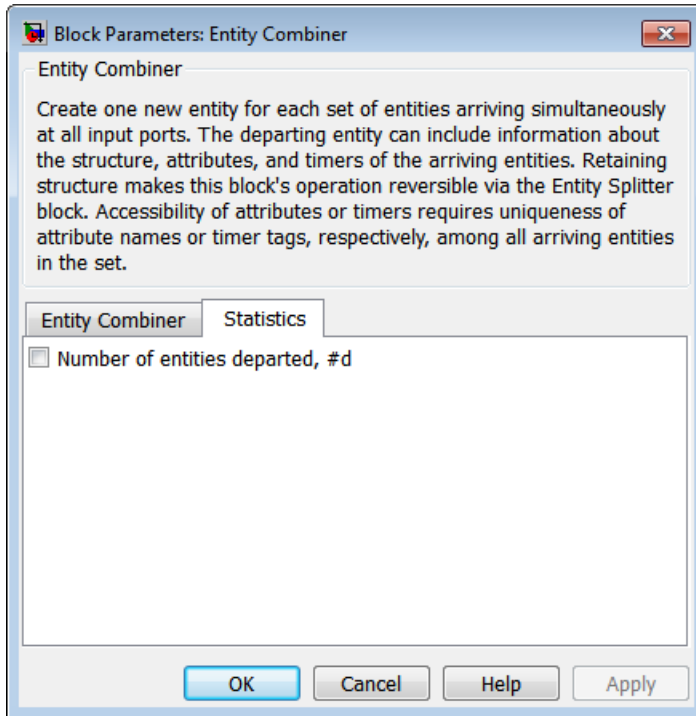
Make attributes accessible in departing entity

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

Make timers accessible in departing entity

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

Statistics Tab



Number of entities departed

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

Examples

- “Wait to Combine Entities”
- “Copy Timers When Combining Entities”
- “Manage Data in Composite Entities”
- Modeling Multicomponent Parts using Combiners and Splitters example

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

“Combine Entities and Allocate Resources”

Entity Departure Counter

Count departures and write result to signal port or attribute

Library

Entity Management



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

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Signal Input Ports

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

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

Entity Output Ports

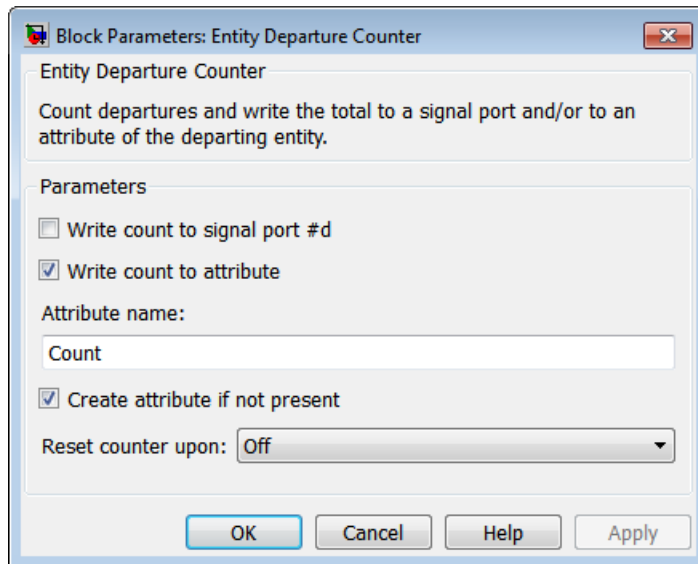
Label	Description
OUT	Port for departing entities.

Signal Output Ports

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

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

Dialog Box



Write count to signal port #d

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

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

Write count to attribute

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

Attribute name

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

Create attribute if not present

Selecting this option enables the block to define a new attribute for the entity count. Otherwise, the block issues an error if the attribute you name in the **Attribute**

name parameter does not exist. You see this field only if select the **Write count to attribute** check box..

Reset counter upon

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

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

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

Resolve simultaneous signal updates according to event priority

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

Event priority

The priority of the reset event, relative to other simultaneous events in the simulation. For details, see “Specify Event Priorities to Resolve Simultaneous Signal Updates”. You see this field only if you set these parameters:

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

Examples

- “Set Attributes”
- “Reset a Counter After a Transient Period”
- “Stop Simulation Upon a Specified Entity Count”

See Also

Instantaneous Entity Counting Scope

“Count Entities”

Entity Departure Function-Call Generator

Convert entity departure event into one or two function calls

Library

Generators/Function-Call Generators



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

Criteria for Generating Function Calls

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

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

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

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Signal Input Ports

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

Entity Output Ports

Label	Description
OUT	Port for departing entities.

Signal Output Ports

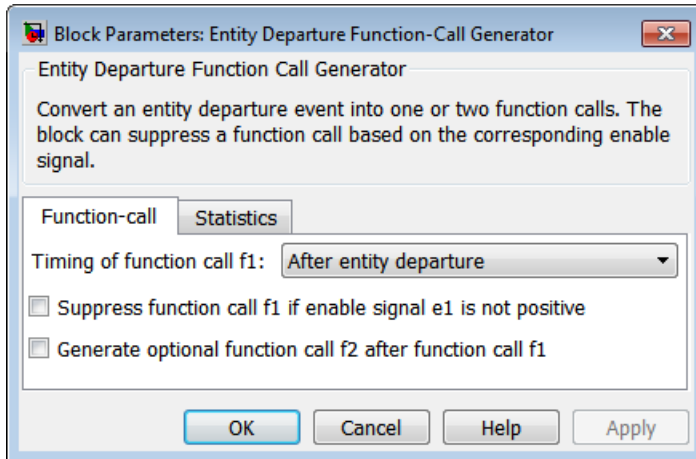
Label	Description	Time of Update When Port Is Present	Order of Update
f1	Function-call signal, possibly contingent on e1 input signal.	Before or after entity departure, depending on Timing of function call f1 parameter	1
f2	Function-call signal, possibly contingent on e2 input signal.	Before entity departure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

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

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

Dialog Box

Function Call Tab

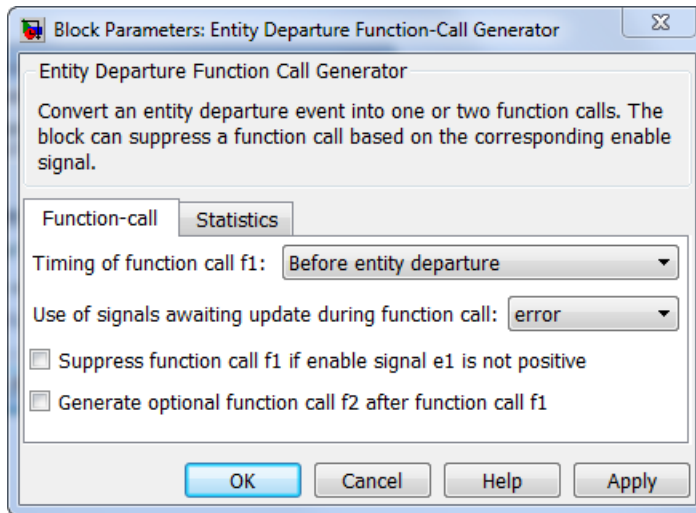


Timing of function call f1

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

Use of signals awaiting update during function call

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



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

Suppress function call f1 if enable signal e1 is not positive

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

Generate optional function call f2 after function call f1

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

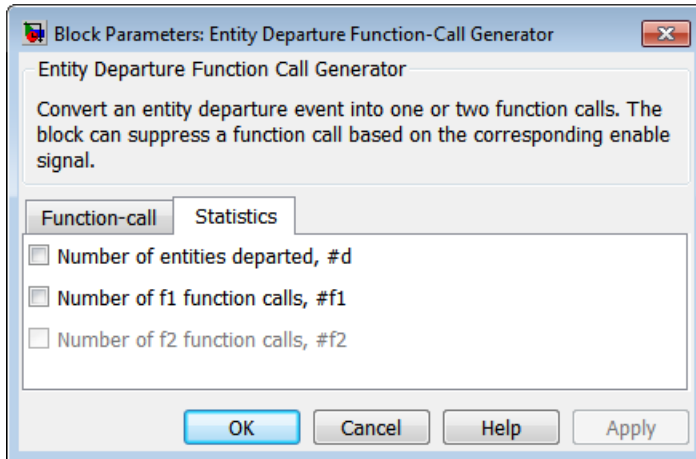
Timing of function call f2

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

Suppress function call f2 if enable signal e2 is not positive

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

Statistics Tab



Number of entities departed

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

Number of f1 function calls

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

Number of f2 function calls

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

Examples

- “Open a Gate Upon Entity Departures”
- Generating Entities as a Markov-Modulated Poisson Process example, within On-Off Modulated Markov Source subsystems

See Also

Signal-Based Function-Call Generator

“Manipulate Events”

Entity Sink

Accept or block entities

Library

SimEvents Sinks

Description



This block provides a way to terminate an entity path:

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

Ports

Entity Input Ports

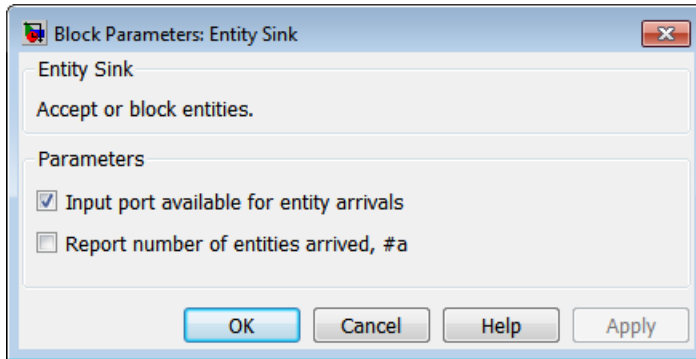
Label	Description
IN	Port for entities that arrive or attempt to arrive.

Signal Output Ports

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

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

Dialog Box



Input port available for entity arrivals

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

Report number of entities arrived, #a

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

Examples

- “Model the Channels”
- “Use an Attribute to Select an Output Port”

See Also

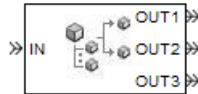
Time-Based Entity Generator, Event-Based Entity Generator

Entity Splitter

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 OUTW port corresponds to the entity that arrived at the INW 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.

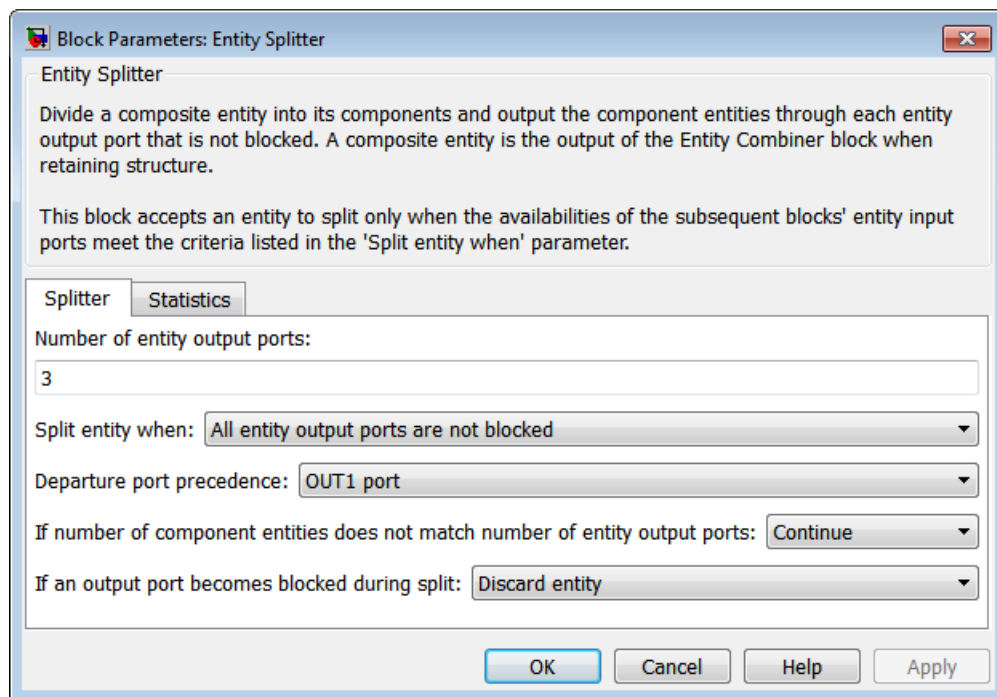
Signal Output Ports

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.

Split entity when

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

Departure port precedence

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

Initial seed

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

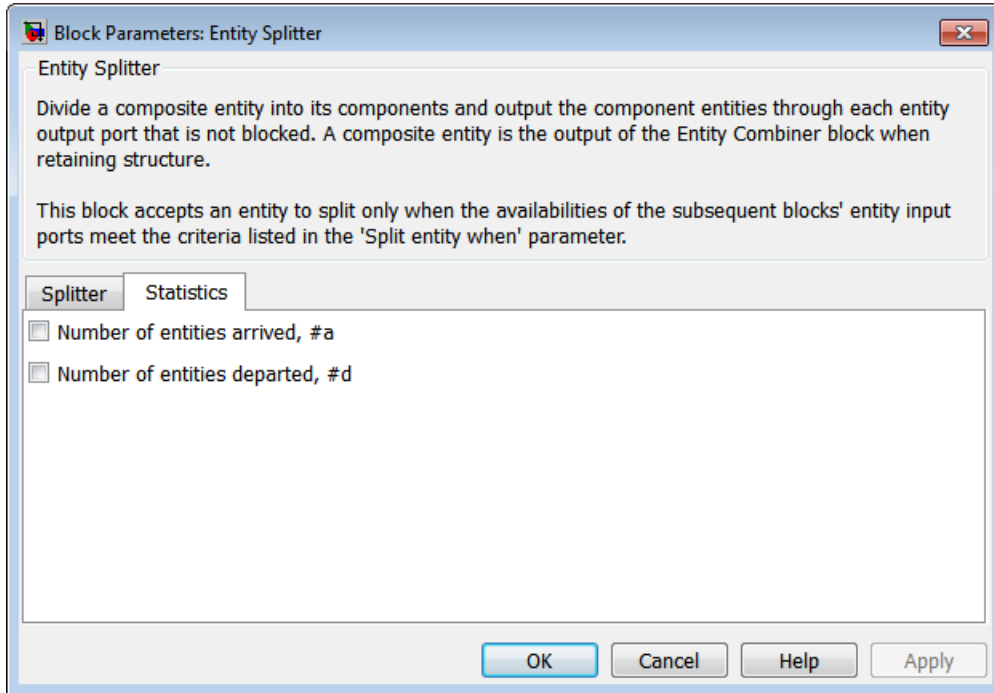
If number of component entities does not match number of entity output ports

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

If an output port becomes blocked during split

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

Statistics Tab



Number of entities arrived

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

Number of entities departed

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

Examples

See “Manage Data in Composite Entities”.

See Also

Entity Combiner

“Combine Entities and Allocate Resources”

Entity-Based Function-Call Event Generator

Generate function call events corresponding to entities

Library

Generators / Event Generators



Note: The Entity-Based Function-Call Event Generator block will be removed in a future release. Use the Entity Departure Function-Call Generator block instead. To update your model to avoid using obsolete blocks, see [seupdate](#).

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

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

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Entity Output Ports

Label	Description
OUT	Port for departing entities.

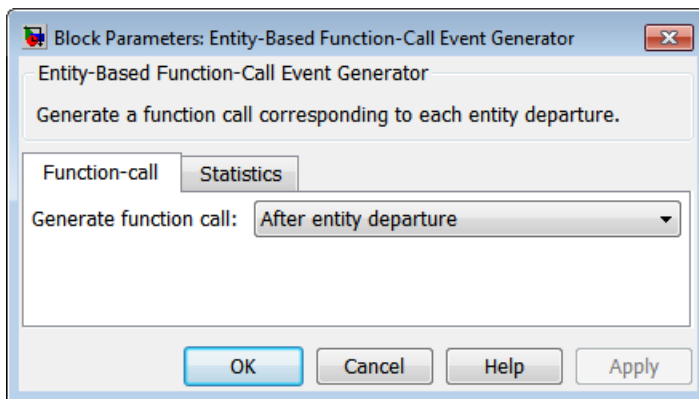
Signal Output Ports

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

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

Dialog Box

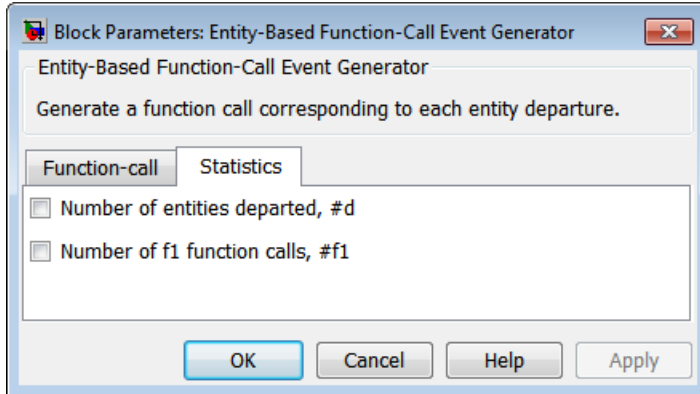
Function Call Tab



Generate function call

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

Statistics Tab



Number of entities departed

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

Number of f1 function calls

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

See Also

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

“Generate Function-Call Events”

Event-Based Entity Generator

Generate entity upon signal-based event or function call

Library

Generators / Entity Generators

Description



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

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

Note: An exceptional case is when the block temporarily suspends its normal entity-generation behavior. See the description of the `Delay first pending entity` option in “Responding to Blockage at the Entity Output Port” on page 2-59.

Responding to Blockage at the Entity Output Port

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

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

Ports

Signal Input Ports

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

Entity Output Ports

Label	Description
OUT	Port through which generated entities depart.

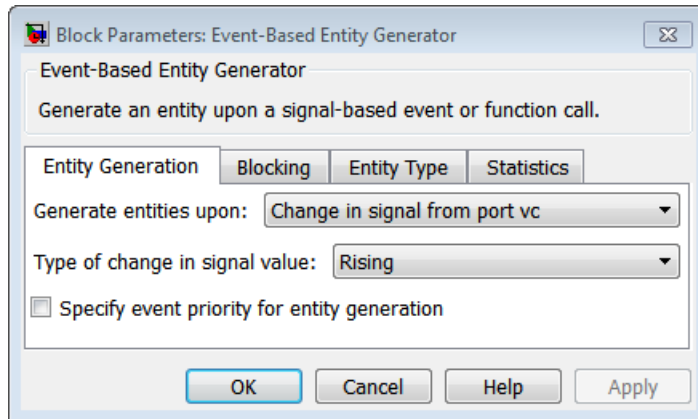
Signal Output Ports

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

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

Dialog Box

Entity Generation Tab



Generate entities upon

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

Trigger type, Type of change in signal value

Trigger type determines whether rising, falling, or either type of trigger edge causes an entity generation. You see this field only if you set **Generate entities upon** to **Trigger from port tr**.

Type of change in signal value determines whether rising, falling, or either type of value change causes an entity generation. You see this field only if you set **Generate entities upon** to **Change in signal from port vc**.

Specify event priority for entity generation

Select this option to prioritize the entity-generation event explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority **SYS1** on the event calendar. For details, see “Resolve Simultaneous Signal Updates”.

Generation event priority

The priority of the entity-generation event, relative to other simultaneous events in the simulation. For details, see “Specify Event Priorities to Resolve Simultaneous

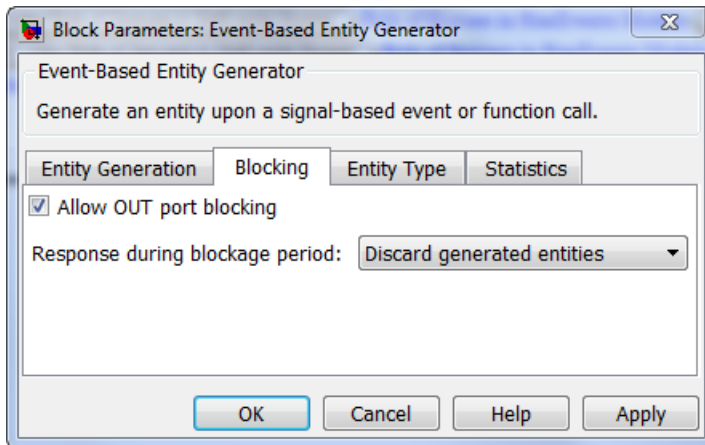
Signal Updates”. You see this field only if you select **Resolve simultaneous signal updates according to event priority**.

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

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

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

Blocking Tab



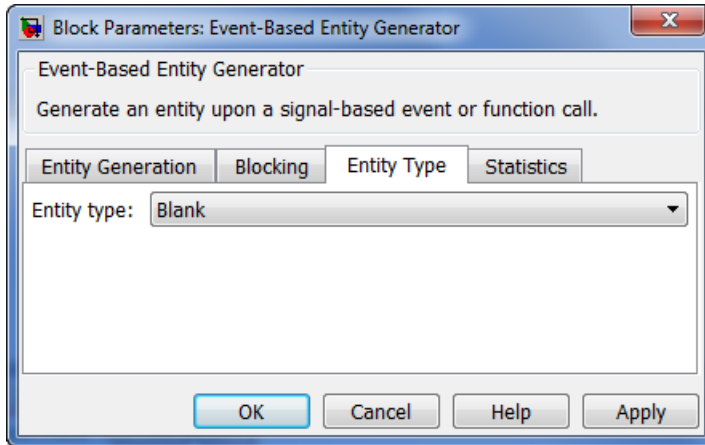
Allow OUT port blocking

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

Response during blockage period

Determines how the block responds if a generated entity cannot depart immediately; see “Responding to Blockage at the Entity Output Port” on page 2-59. You see this field only if you select **Allow OUT port blocking**.

Entity Type Tab

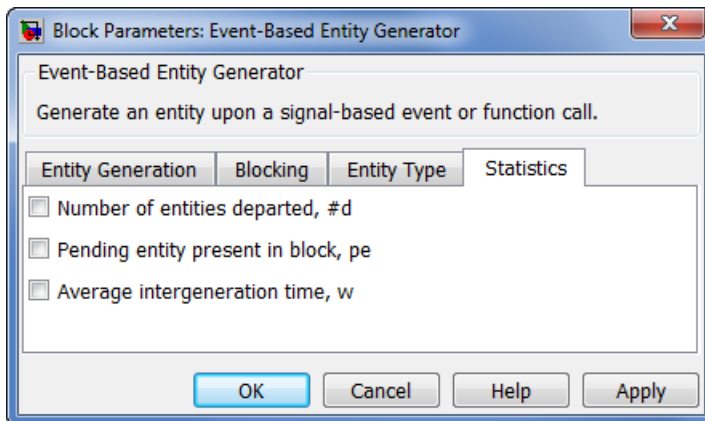


Entity type

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

Statistics Tab

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



Number of entities departed

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

Pending entity present in block

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

Average intergeneration time

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

Examples

- “Plot Event Counts to Check for Simultaneity”
- “Choose Values for Event Priorities”
- “Sample Use Cases for Event-Based Generation of Entities”

See Also

Time-Based Entity Generator

Entity Sink

“Generate Entities When Events Occur”

Event-Based Random Number

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

Library

Generators / Signal Generators



Description

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

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

Connecting to Other Blocks

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

The direct or indirect connections to signal input ports must have the multiplicities in the table.

Type of Signal Input Port	Exact Number of Connections
“Notifying Ports”	1
“Reactive Ports”	0
	Tip To create a random signal that can be an input to a reactive port, use the

Type of Signal Input Port	Exact Number of Connections
	procedure in “Generate Random Signals Based on Arbitrary Events”.
“Monitoring Ports”	Arbitrary number
Ports of blocks that appear in “Sink Blocks”	Arbitrary number

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

- Appears in “Computational Blocks”
- Has exactly one input signal
- Has no function-call output signals

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

Distribution Types

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

Distribution	Additional Parameters
Exponential	Mean
Uniform	Minimum, Maximum
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

Distribution	Additional Parameters
Lognormal	Threshold, Mu, Sigma
Log-logistic	Threshold, Scale
Beta	Minimum, Maximum, Shape parameter a, Shape parameter b
Discrete uniform	Minimum, Maximum, Number of values
Weibull	Threshold, Scale, Shape
Arbitrary continuous	Value vector, Cumulative probability function vector
Arbitrary discrete	Value vector, Probability vector

For information about the definitions and properties of each distribution, see “References” on page 2-76 below.

Range of Output Values

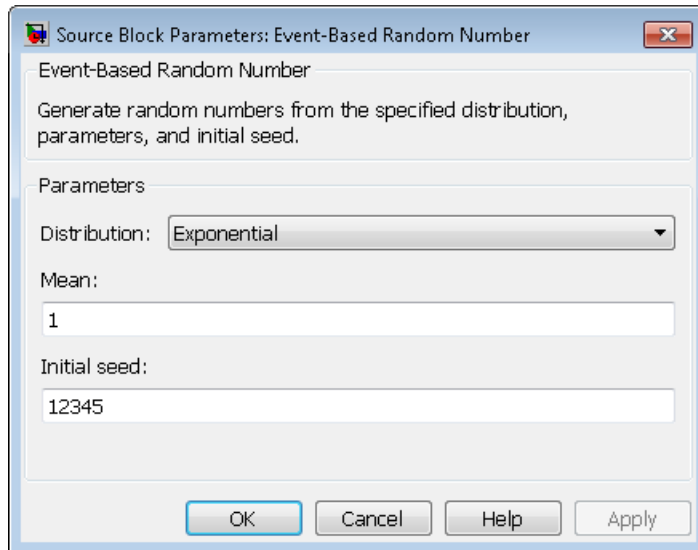
Different distributions have different output ranges. Make sure the distribution and parameters you choose are suitable for your application. For example, when generating random service times, do not use a Gaussian distribution because it can produce negative numbers.

Ports

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

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

Dialog Box



Distribution

The distribution from which the block generates random numbers.

Mean

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

Minimum

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

Maximum

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

Probability for output to be 1

The probability of a one in a Bernoulli distribution.

Probability of success in a single trial

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

Number of trials

The number of trials used to describe a binomial distribution.

Mode

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

Threshold, Scale, Shape

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

Threshold, Mu, Sigma

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

Standard deviation

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

Shape parameter a, Shape parameter b

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

Number of values

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

Value vector

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

Cumulative probability function vector

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

Probability vector

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

Initial seed

A nonnegative integer that initializes the random number generator.

Examples

- “Examples of Random Event-Based Signals”

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 and Machine Learning 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 and Machine Learning Toolbox software.

Bernoulli Distribution

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

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

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

Binomial Distribution

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

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

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

Triangular Distribution

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

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

Gamma Distribution

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

where θ is the **Threshold** parameter, b is the **Scale** parameter, and a is the **Shape** parameter. The **Scale** and **Shape** parameters must be positive. Also, Γ is the gamma function (`gamma` in MATLAB code).

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

Gaussian (Normal) Distribution

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

where μ is the **Mean** parameter and σ is the **Standard deviation** parameter. The standard deviation parameter must be nonnegative.

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

Geometric Distribution

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

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

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

In the special case where the **Probability of success in a single trial** parameter is 1, then

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

This is a discrete distribution.

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

Poisson Distribution

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

where λ is the **Mean** parameter, a positive number. This is a discrete distribution.

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

Lognormal Distribution

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

where θ is the **Threshold** parameter, μ is the **Mu** parameter, and σ is the **Sigma** parameter. The **Sigma** parameter must be positive.

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

Log-Logistic Distribution

The log-logistic distribution is derived from the logistic distribution, as follows:

X = Random variable with logistic distribution

$Y = e^X$ = Random variable with log-logistic distribution

The probability density function for the logistic distribution is

$$f_{\text{logistic}}(x) = \frac{1}{b} \cdot \frac{e^{(x-\theta)/b}}{\left(1 + e^{(x-\theta)/b}\right)^2}$$

where θ is the **Threshold** parameter and b is the **Scale** parameter. The **Scale** parameter must be positive.

Beta Distribution

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

where L is the **Minimum** parameter, M is the **Maximum** parameter, a is the **Shape parameter a** parameter, b is the **Shape parameter b** parameter, and $B(a,b)$ is the beta function defined by

$$B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1} dt$$

The two shape parameters must be positive.

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

Discrete Uniform Distribution

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

where L is the **Minimum** parameter, U is the **Maximum** parameter, and K is the **Number of values** parameter. This is a discrete distribution. If $(U-L)/(K-1)$ and L are both integers, then all outputs from this distribution are integers.

Similar functions are `randi` in MATLAB software and `unidrnd` in the Statistics and Machine Learning Toolbox software.

Weibull Distribution

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

where θ is the **Threshold** parameter, α is the **Scale** parameter, and γ is the **Shape** parameter. The **Scale** and **Shape** parameters must be positive.

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

References

- [1] Evans, M., N. Hastings, and B. Peacock. *Statistical Distributions*. Wiley-Interscience, 2000.
- [2] Johnson, N. L., S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, Volume 1. Wiley-Interscience, 1993.
- [3] Johnson, N. L., S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, Volume 2. Wiley-Interscience, 1994.
- [4] Johnson, N. L., S. Kotz, and A. W. Kemp. *Univariate Discrete Distributions*. Wiley-Interscience, 1993.

See Also

Signal Latch, Event-Based Sequence

“Generate Random Signals”

Event-Based Sequence

Generate sequence of numbers from specified column vector

Library

Generators / Signal Generators

Description

This block generates an event-based signal using data you provide, inferring from a subsequent block when to output the next value from your data. You specify the data as a column vector using the **Vector of output values** parameter. The parameter value can be any MATLAB language expression that evaluates to a column vector, including the name of a column vector variable in the MATLAB base workspace. As an example of inferring timing from a subsequent block, if you connect this block to the **t** input port of a Single Server block, then the Event-Based Sequence block outputs a new value each time an entity arrives at the server.

Behavior After Data Runs Out

If the block needs more data than the vector contains, subsequent output values follow a rule you specify using the **Form output after final data value by** parameter. The table below lists possible values for this parameter.

Note: In all cases, the choice of parameter value affects only the values, not the timing, of the output signal. The output signal is always an event-based signal whose sample time hits depend on notifications from a subsequent block.

Parameter Value	Description
Cyclic repetition	When the block needs a new output value after exhausting the data, it starts over at the beginning of the vector.

Parameter Value	Description
Holding final value	After exhausting the data, the block outputs the last data value for every sample time hit.
Setting to infinity	After exhausting the data, the block outputs the value <code>inf</code> for every sample time hit. For example, if the block outputs intergeneration times for an entity generator, then the generator produces up to a fixed number of entities.
Setting to zero	After exhausting the data, the block outputs zero for every sample time hit. For example, if the block outputs service times for a server, then the server delays up to a fixed number of entities.

Connecting to Other Blocks

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

The direct or indirect connections to signal input ports must have the multiplicities in the table.

Type of Signal Input Port	Exact Number of Connections
“Notifying Ports”	1
“Reactive Ports”	0
	Tip To create a sequence that can be an input to a reactive port, use the procedure in “Generate Sequences Based on Arbitrary Events”. Alternatively, use time-based blocks, such as Repeating Sequence Stair and From Workspace, followed by the Timed to Event Signal block.
“Monitoring Ports”	Arbitrary number
Ports of blocks that appear in “Sink Blocks”	Arbitrary number

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

- Appears in “Computational Blocks”
- Has exactly one input signal
- Has no function-call output signals

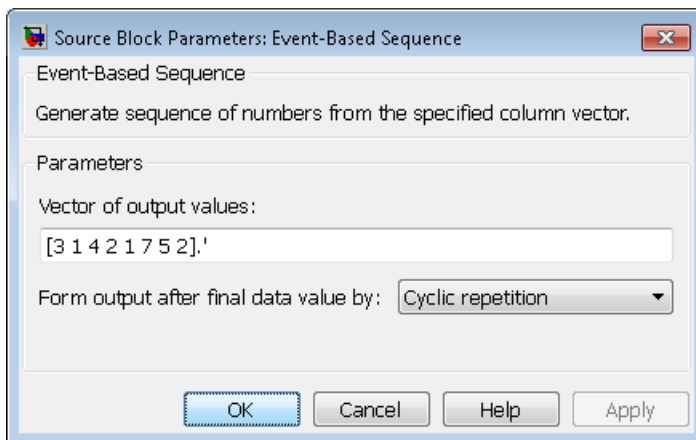
Tip For an indirect connection to the Atomic Subsystem block, the restrictions on input and output signals apply to the subsystem itself, not the blocks inside the subsystem.

Ports

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

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

Dialog Box



Vector of output values

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

Form output after final data value by

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

Examples

- “Specify Generation Times for Entities”
- “Count Simultaneous Departures from a Server”
- “Set Attributes”

See Also

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

“Create Event-Based Signals Using Data Sets”

Event Filter

Conditionalize, suppress, or prioritize execution of Atomic Subsystem

Library

SimEvents Ports and Subsystems



Description Event Filter

This block influences an Atomic Subsystem block by specifying the signal-based events upon which the subsystem executes. This block can also prioritize the execution of a subsystem with regard to other events occurring simultaneously by scheduling a subsystem execution on the event calendar. Consider an event-based signal that is an input to an Atomic Subsystem block. Without the Event Filter block, every sample time hit of the signal causes the subsystem to execute immediately. Inserting the Event Filter block on that signal line enables you to influence the subsystem behavior as follows:

- Specify the type of signal-based event that causes the subsystem to execute. Choices are:
 - Sample time hit
 - Change in signal value (rising, falling, or either)
 - Trigger (rising, falling, or either)

If the input signal of this block is a nonscalar array, the block detects one qualifying event if any of the positions in the array has a qualifying event. For example, a change in signal value from [1 2 3] to [1 5 6] represents one qualifying event, not two. If N distinct qualifying events occur at distinct sample time hits in the input signal of this block, the subsystem executes N times and updates its output signals N times.

- Prevent the input signal of this block from causing the subsystem to execute. In this case, the signal passively provides data to the subsystem. The subsystem can still execute based on signal-based events of a different input signal.

- Prioritize the subsystem execution, relative to other simultaneous events in the simulation. Instead of occurring immediately upon a signal-based event, the execution becomes a scheduled event on the event calendar.

Connecting to Other Blocks

The output port of this block can connect to only one input port of an Atomic Subsystem block. The connection line cannot branch.

Behavior During Simulation

When the input signal of an Event Filter block has a sample time hit, it does the following:

- 1 Updates its output signal with the value of the input signal. This value is available to the Atomic Subsystem block to which the Event Filter block connects.
- 2 Determines whether to execute the Atomic Subsystem block, based on the settings in the block dialog box of the Event Filter block. If the Event Filter block is not supposed to execute the Atomic Subsystem block, the Event Filter does nothing further, until the next sample time hit of the input signal. Otherwise, processing continues to the next step.
- 3 Determines when to execute the Atomic Subsystem block.
 - If you did not select the **Resolve simultaneous signal updates according to event priority** option, the Event Filter block executes the Atomic Subsystem block immediately.
 - If you select the **Resolve simultaneous signal updates according to event priority** option, the Event Filter block schedules an event on the event calendar. The event time is the current simulation time. The event priority is the value of the **Event priority** parameter in the Event Filter block. When the event calendar executes this event, the Atomic Subsystem block performs its computation.
 - If you select both the **Resolve simultaneous signal updates according to event priority** option, and the configuration parameter **Prevent duplicate events on multiport blocks and branched signals**, the software uses the **Event priority** parameter to help Simulink to sort blocks in the model. In this case, the software no longer schedules an event on the event calendar.

Ports

Signal Input Ports

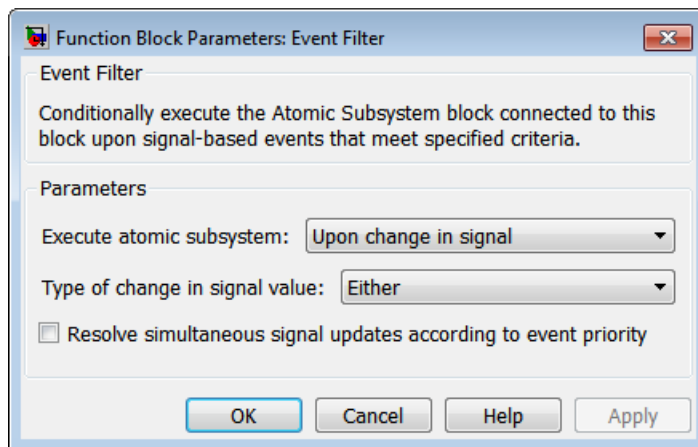
Label	Description
None	Event-based signal. The signal can have any fixed dimension or complexity. It has data type double.

Signal Output Ports

Label	Description
None	Event-based signal whose value matches that of the input signal. This output signal connects to an input port of an Atomic Subsystem block. When the Event Filter block detects a qualifying signal-based event in its input signal, the subsystem executes immediately or the block schedules an execution event on the event calendar. (Block parameters determine the type of event that qualifies and the choice of immediate or scheduled execution.)

The initial output value is the same as that of the input signal. This value is in effect before the first sample time hit of the input signal.

Dialog Box



Execute atomic subsystem

Determines what constitutes a qualifying event in the input signal of this block. If the signal is complex, you must select `Upon sample time hit` or `Never`.

Trigger type

The type of trigger that further restricts the event type specified in **Execute atomic subsystem**. You see this field only if you set **Execute atomic subsystem** to `Upon trigger`.

Type of change in signal value

The type of change in the signal value that further restricts the event type specified in **Execute atomic subsystem**. You see this field only if you set **Execute atomic subsystem** to `Upon change in signal`.

Resolve simultaneous signal updates according to event priority

Select this option to control the sequencing of the subsystem execution in response to updates in the input signal of this block, relative to other simultaneous events in the simulation. If you do not select this option, the application executes the subsystem immediately upon detecting the signal-based event. For details, see “Resolve Simultaneous Signal Updates”.

Event priority

The priority of the subsystem execution event (in response to updates in the input signal of this block), relative to other simultaneous events in the simulation. For details, see “Specify Event Priorities to Resolve Simultaneous Signal Updates”.

Use of this parameter depends on the following:

- You see this field only if you select **Resolve simultaneous signal updates according to event priority**.
- If you also select the configuration parameter **Prevent duplicate events on multiport blocks and branched signals** in your model, the software uses the **Event priority** parameter to help Simulink to sort blocks in the model. In this case, the software does not schedule an event that you can view on the SimEvents event calendar.

Examples

- “Reset an Average Periodically”
- “Observe Service Completions”
- “Effects of Specifying Event Priorities”

See Also

“Suppress Computations by Filtering Out Events”

“Perform Computations in Atomic Subsystems”

“Resolve Simultaneous Signal Updates”

Event to Timed Signal

Convert event-based signal to time-based signal

Library

Gateways

Description

This block converts an event-based data signal into a time-based data signal. The output signal assumes exactly one value at any given time on the simulation clock. The output signal is almost identical to the input signal, except:

- The output signal omits zero-duration values, if any, from the input signal.
- The output signal has a sample time type of “fixed in minor step.” As a result, the output signal might have sample time hits at times unrelated to the input signal but related to other time-based signals in the model.
- The output signal is suitable for modeling time-based dynamics. The signal cannot be an input to a block that requires an event-based input signal. Blocks that can process either time-based or event-based signals might process them differently.
- The initial output value is the same as the initial input value. However, if the input signal is undefined at $T = 0$, as in the case of an atomic subsystem that has an event-based input signal, the output signal of this block has an initial output of 0.

Ports

Signal Input Ports

Label	Description
None	Event-based signal. The signal can have any fixed dimension, complexity, or data type.

Signal Output Ports

Label	Description
None	Time-based signal

See Also

Timed to Event Signal

“Time-Based and Event-Based Signal Conversion”

Event to Timed Function-Call

Convert event-based function call to time-based function call

Library

Gateways

Description

This block converts a scalar event-based function call into a time-based function call. The output signal is almost identical to the input signal, except that the output can be an input to a block that requires a time-based function-call input signal.

Ports

Signal Input Ports

Label	Description
None	Event-based function-call signal.

Signal Output Ports

Label	Description
None	Time-based function-call signal.

See Also

Timed to Event Function-Call

“Time-Based and Event-Based Signal Conversion”

FIFO Queue

Store entities in sequence for undetermined length of time

Library

Queues



This block stores up to N entities simultaneously, where N is the **Capacity** parameter value. The block attempts to output an entity through the **OUT** port, but retains the entity if the **OUT** port is blocked. If the block is storing multiple entities and no entity times out, then entities depart in a first-in, first-out (FIFO) fashion. If an entity in this block is scheduled to time out, then it might depart prematurely via the optional **TO** port. For an example, see “Use Timeouts to Limit Entity Queueing Time”. The length of time that an entity stays in this block cannot be determined in advance.

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

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities, which are stored.

Entity Output Ports

Label	Description
OUT	Port for departing entities that do not time out while in this block.

Label	Description
TO	Port for entities that time out while in this block. You see this port only if you select Enable TO port for timed-out entities . This port must not be blocked when an entity attempts to depart here.

Signal Output Ports

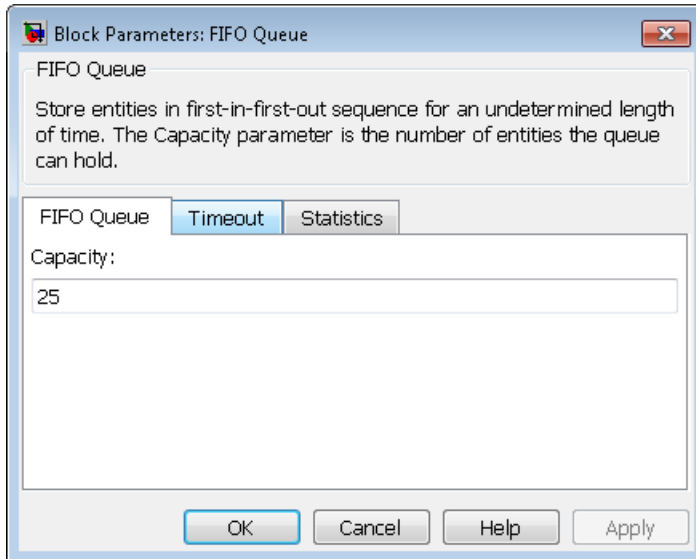
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block via the OUT port since the start of the simulation.	After entity departure via the OUT port	3
#n	Number of entities currently in the queue.	After entity arrival and after entity departure	2
w	Sample mean of the waiting times in this block for all entities that have departed via any port.	After entity departure	1
len	Average number of entities in the queue over time, that is, the time average of the #n signal.	After entity arrival and after entity departure.	1
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the TO port	3

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

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

Dialog Box

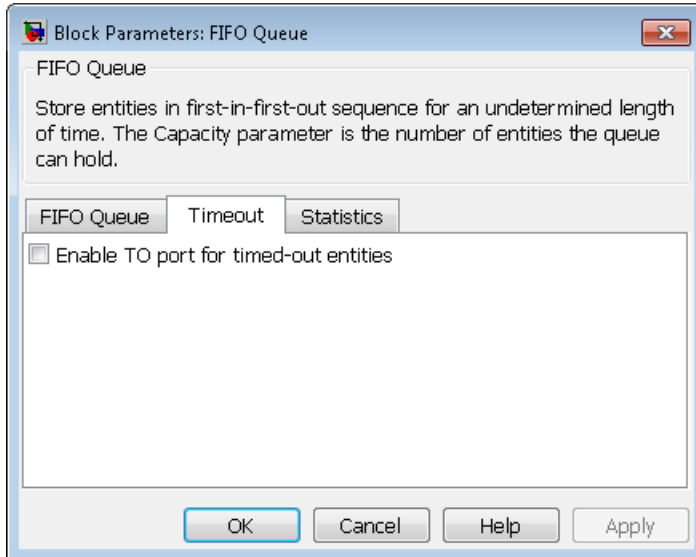
FIFO Queue Tab



Capacity

Determines how many entities the block can store at a time. The capacity must be a positive integer or Inf.

Timeout Tab

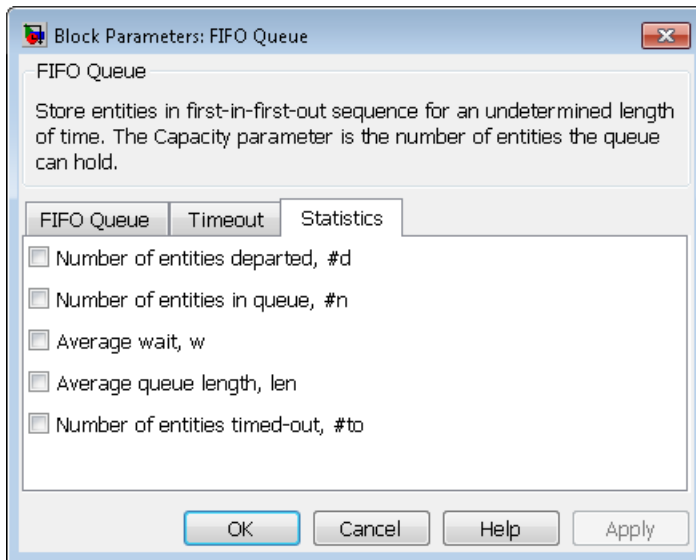


Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the Schedule Timeout block.

Statistics Tab

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



Number of entities departed

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

Number of entities in queue

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

Average wait

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

Average queue length

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

Number of entities timed out

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

Examples

- “Build a Discrete-Event Model”
- “Select the First Available Server”
- “Round-Robin Approach to Choosing Inputs”

- “Constructs Involving Queues and Servers”
- “Example of a Logical Queue”
- “LIFO Queue Waiting Time”

See Also

LIFO Queue, Priority Queue

“Queues in SimEvents Models”

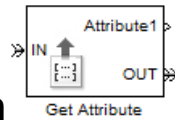
Get Attribute

Output value of entity attribute

Library

Attributes

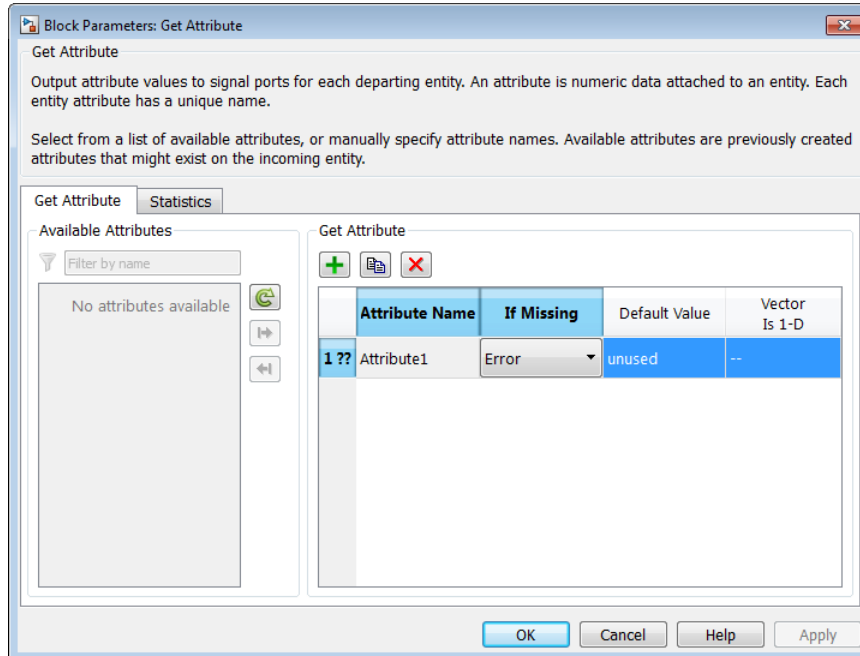
Description



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

Dialog Box

Get Attribute Tab



Available Attributes




Use the **Available Attributes** controls to:

- Select the attributes from incoming entity paths that you want to access.
- Add the attributes to the **Get Attribute** table, where you can modify them.

The list displays all the attributes on all the incoming entities. (If the entity paths entering the Get Attribute block do not have any attributes, the **Available Attributes** list is empty).

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

Use the buttons in the **Available Attributes** section to help build the attributes table. The buttons perform these actions:

Button	Action
	Refresh the Available Attributes list. This action updates the list with any upstream model changes you make while the block dialog box is open.
	Add the selected attribute to the Get Attribute table.
	Move the selected attribute from the Get Attribute table to the Available Attributes list. Note: If the selected attribute is one you added manually, this button appears dimmed.

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

Message	Meaning
> Attribute already selected	You have already added the attribute to the Get Attribute table. You cannot add the attribute to the table again.
* Attribute may not be present	When multiple entity paths enter the block, all entities might not have the same attributes. Attributes that are not on all entering entities display an asterisk in the list, and this message appears. If you add such an attribute to the Get Attribute table, the behavior depends on how the If Missing field is set.

Get Attribute




Use the controls under **Get Attribute** to build and manage the list of attributes to access on each incoming entity. Each attribute appears as a row in a table.

Using these controls, you can:

- Specify an attribute manually to access on the entity.

- Modify an attribute that you added to the table from the **Available Attributes** list to access on the entity.

The buttons under **Get Attribute** perform these actions:

Button	Action	Notes
	Add a template attribute to the table.	Rename the attribute and specify its properties.
	Add a copy of the selected attribute to the table to use as the basis of a new attribute.	Rename the copy. Two attributes cannot have the same name.
	Remove the selected attribute from the Get Attribute table.	When you delete an attribute this way, no confirmation appears and you cannot undo the operation.

Note: If you delete a row and apply the change, the deletion can affect signal output ports corresponding to other attributes. For example, if the block has a signal output port **A2** and you delete the attribute with a port marked **A1**, the block relabels **A2** as **A1**. Verify that any signal that connects to the relabeled port is still connected as you expect.

The table displays the attributes you added from the **Available Attributes** list or added manually. Use it to set these four attribute properties:

Property	Specify	Use
Attribute Name	<p>The name of the attribute to access. Each attribute must have a unique name.</p> <p>If the attribute name does not match an attribute listed in Available Attributes, the block displays ?? next to the attribute name. This symbol denotes that the attribute is not present on incoming</p>	Double-click the existing name, and then type the new name.

Property	Specify	Use
	entity paths. The simulation behavior you see depends on the value of If Missing (see “Missing Attributes” on page 2-99).	
If Missing	The response of the block when the entity does not have an attribute named in the table.	Select a block response from the list.
Default Value	The value for the corresponding output signal if the entity does not have an attribute specified in the table. To learn about the kind of data you can use as a default value, see “Attribute Value Support”. You can set this field only if you set If Missing to Default value or Warn .	Double-click the field and type a value.
Vector Is 1-D	Whether the block considers the default value as a vector of length N when Default Value evaluates to an N-element row or column vector. Otherwise, the block considers the default value as a multidimensional array. This option affects attributes whose If Missing parameter is set to Default value or Warn .	Select the check box to treat the attribute as a vector of length N. Clear it to treat the attribute as a multidimensional array.

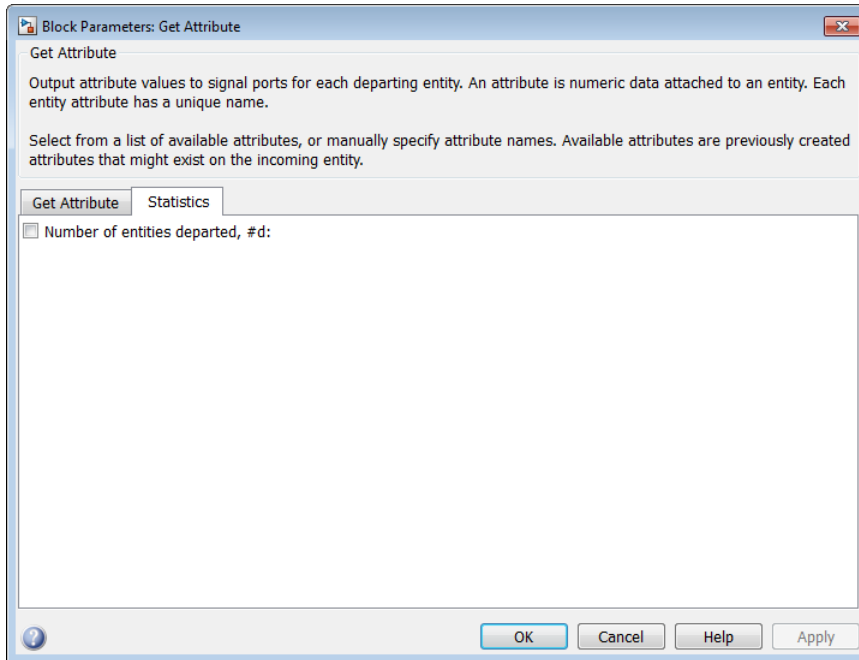
Missing Attributes

You can specify the block behavior if the arriving entity does not have an attribute listed in the table of the block dialog box. Use the **If Missing** parameter for that attribute.

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

Statistics Tab

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



Number of entities departed

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

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities

Entity Output Ports

Label	Description
OUT	Port for departing entities

Signal Output Ports

Label	Description	Time of Update When Statistic Is On
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure
<i>Attribute name</i>	Value of the attribute of the same name specified in the table. The default Name that corresponds to each row is Attributex , where x = 1, 2, 3, etc.	After entity departure

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

Examples

- “Lesson 3: Add Event-Based Behavior”
- “Use Block Diagrams to Manipulate Attributes”

See Also

Set Attribute

“Manipulate Entity Attributes”

Infinite Server

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. For an example that uses the **TO** port of a queue block in the same way, see “Use Timeouts to Limit Entity Queueing Time”.

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”.

The **IN** port of an infinite server is always available. You can interpret an infinite server as a mechanism for delaying entities. Some discussions of this block suggest this interpretation by using the word *delay* instead of *serve*.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities, which will be served.

Signal Input Ports

Label	Description
t	Service time, in seconds, for a newly arrived entity. This signal must be an event-based signal. You see this port only if you set Service time from to Signal port t .

Entity Output Ports

Label	Description
OUT	Port for departing entities that have completed their service time and have not timed out while in this block.
TO	Port for entities that time out while in this block. You see this port only if you select Enable TO port for timed-out entities . This port must not be blocked when an entity attempts to depart here.

Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block via the OUT port since the start of the simulation.	After entity departure via the OUT port	5
#n	Number of entities in the block.	After entity arrival and after entity departure	4
pe	A value of 1 indicates that the block stores at least one entity that has tried and failed to	After the block stores an entity that has tried and failed to	1

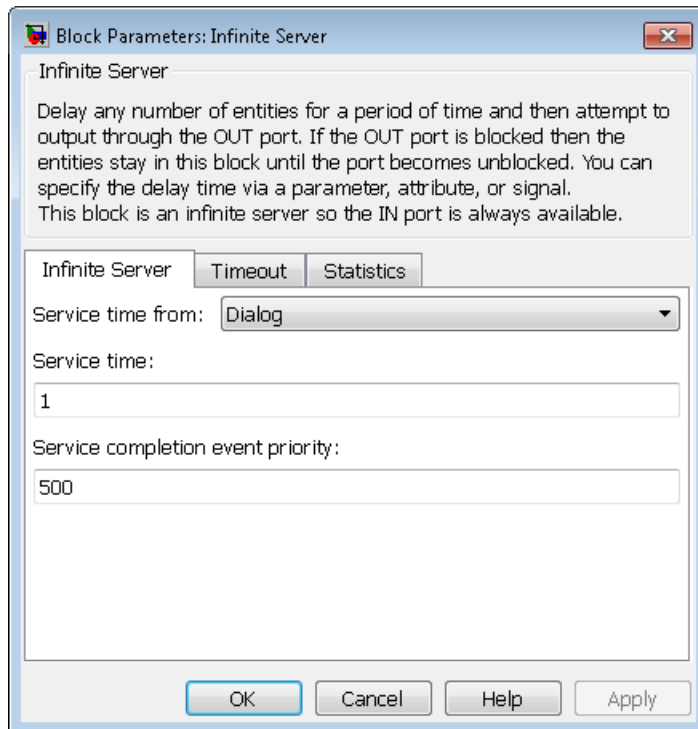
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
	depart. Such entities are pending entities. A value of 0 indicates that the block does not store any pending entities.	depart. In this case, the signal value is 1. After the departure of a pending entity. In this case, the signal value depends on whether any other pending entities remain in the block.	
#pe	Number of pending entities in the block.	After the block stores an entity that has tried and failed to depart. After the departure of a pending entity.	3
w	Sample mean of the waiting times in this block for all entities that have departed via any port. An entity's waiting time might exceed its service time if the OUT port is blocked when the entity completes service.	After entity departure	2
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the TO port	5

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

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

Dialog Box

Infinite Server Tab



Service time from

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

Service time

The service time, in seconds, for all entities. You see this field only if you set **Service time from** to **Dialog**.

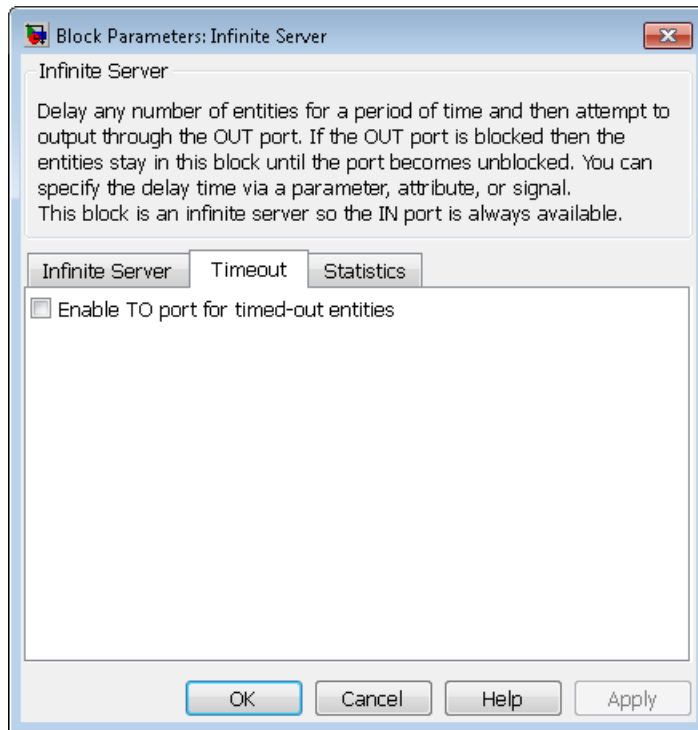
Attribute name

The name of the attribute whose value the block uses as the service time for an entity. You see this field only if you set **Service time from** to **Attribute**.

Service completion event priority

The priority of the service completion event, relative to other simultaneous events in the simulation.

Timeout Tab

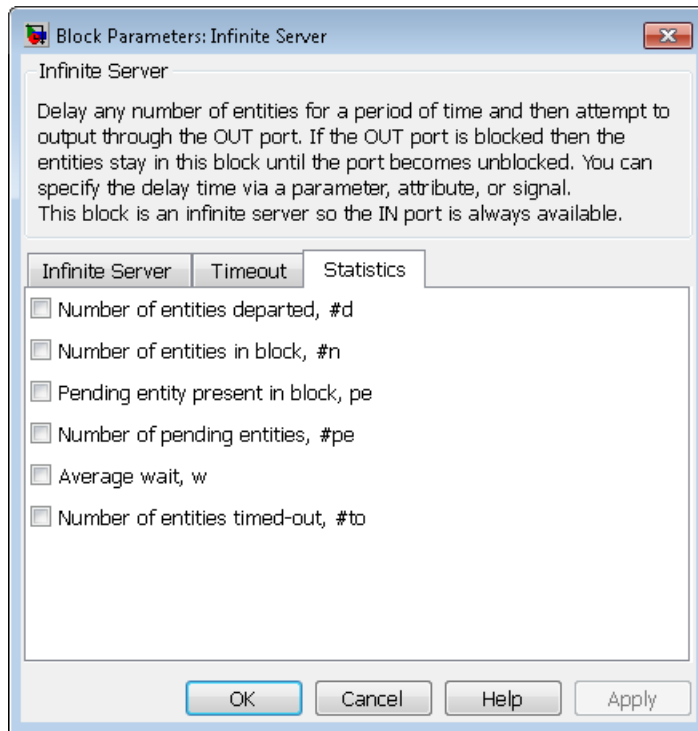


Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the Schedule Timeout block.

Statistics Tab

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



Number of entities departed

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

Number of entities in block

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

Pending entity present in block

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

Number of pending entities

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

Average wait

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

Number of entities timed out

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

Examples

- “Restart a Timer from Zero”
- “Count Simultaneous Departures from a Server”

See Also

Single Server, N-Server

“Servers in SimEvents Models”

Initial Value

Output specified value until first sample time hit

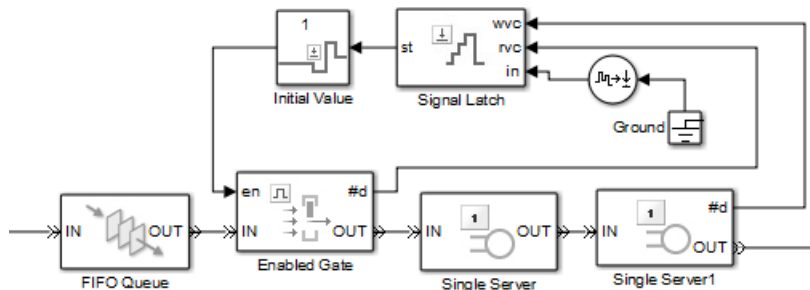
Library

Signal Management



This block establishes an initial value for an event-based signal. Before the first sample time hit at the input port, the value of the output signal is the **Value until first sample time hit** parameter value. Starting from the first sample time hit, the output signal is identical to the input signal.

The following model fragment illustrates block usage in a feedback loop. When the simulation starts, the Initial Value block provides an initial value of 1 that opens the gate to permit the first entity to advance into the feedback loop. Without a nonzero initial value, no entity would arrive at the servers and the Signal Latch block would never experience any events.



Note: The IC block in the Simulink library set operates in a time-based manner and is not suitable for event-based signals.

Ports

Signal Input Ports

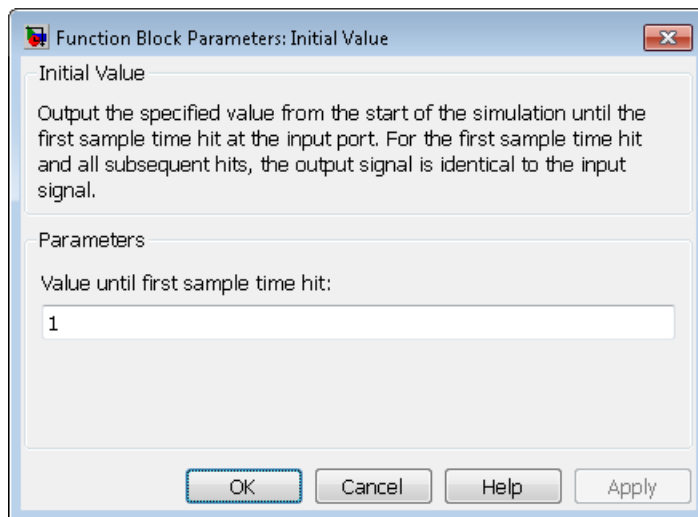
Label	Description
None	The first sample time hit in this signal causes the block to stop using the initial value from the block dialog box. From then on, the output signal is identical to the input signal. This signal must be an event-based signal.

Signal Output Ports

Label	Description
None	The value is either the initial value in the block dialog box or the input signal value, depending on whether the input signal has had a sample time hit yet during the simulation.

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

Dialog Box



Value until first sample time hit

The value to output before the first sample time hit of the input signal. The value of this parameter must have the same dimensions, data type, and complexity as the input signal.

Examples

- “Control Joint Availability of Two Servers”
- “Failure and Repair of a Server”
- “Limit the Time Until Service Completion”

See Also

IC, Signal Latch

“Specify Initial Values of Event-Based Signals”

Input Switch

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 signal must be an event-based signal. You see this port only if you set Switching criterion to From signal port p.

Entity Output Ports

Label	Description
OUT	Port for departing entities.

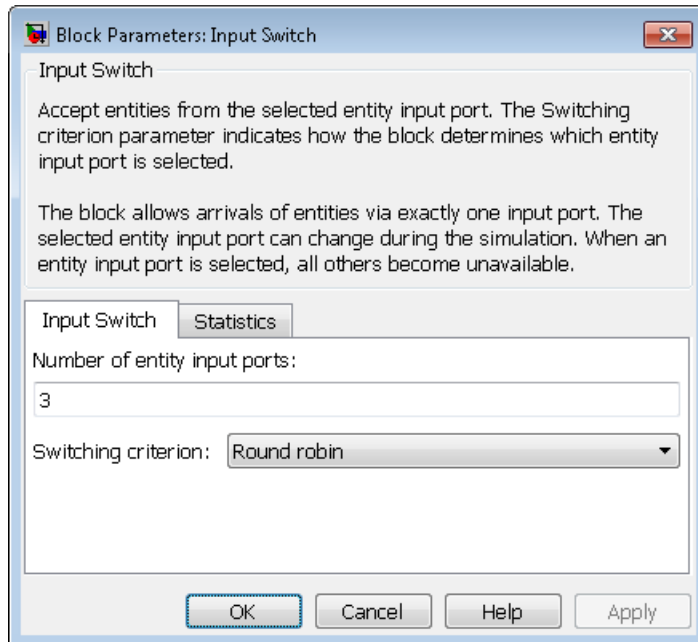
Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	2
last	Index of the input port that was available the last time an entity departed. The initial value is 0. After an entity has departed, values are 1, 2, 3,..., Number of entity input ports .	After entity departure	1

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

Dialog Box

Input Switch Tab



Number of entity input ports

Determines how many entity input ports the block has.

Switching criterion

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

Initial seed

A nonnegative integer that initializes the random number generator used to select an entity input port. You see this field only if you set **Switching criterion** to Equiprobable.

Resolve simultaneous signal updates according to event priority

Select this option to prioritize the port-selection event explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has

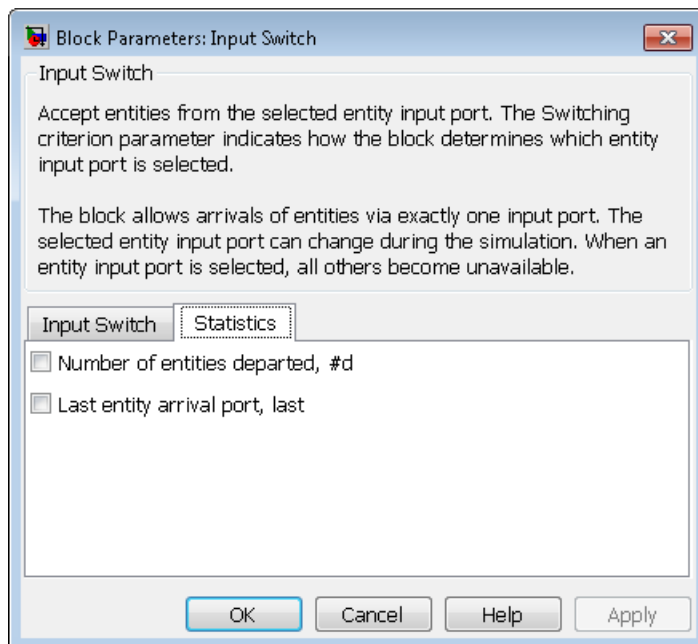
priority **SYS1** on the event calendar. For details, see “Resolve Simultaneous Signal Updates”. You see this field only if you set **Switching criterion** to **From signal port p**.

Event priority

The priority of the port-selection event, relative to other simultaneous events in the simulation. For details, see “Specify Event Priorities to Resolve Simultaneous Signal Updates”. You see this field only if you set **Switching criterion** to **From signal port p** and select **Resolve simultaneous signal updates according to event priority**.

Statistics Tab

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



Number of entities departed

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

Last entity arrival port

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

Examples

- “Choose Values for Event Priorities”
- “Round-Robin Approach to Choosing Inputs”
- “Compound Switching Logic”

See Also

Output Switch

“Select Arrival Path Using Input Switch”

Instantaneous Entity Counting Scope

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.

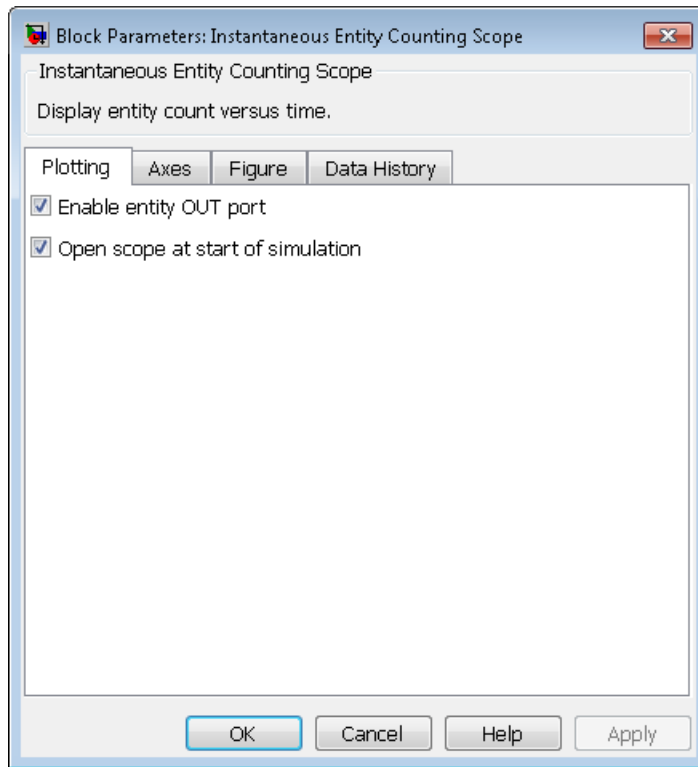
Entity Output Ports

Label	Description
OUT	Port for departing entities. You see this port 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



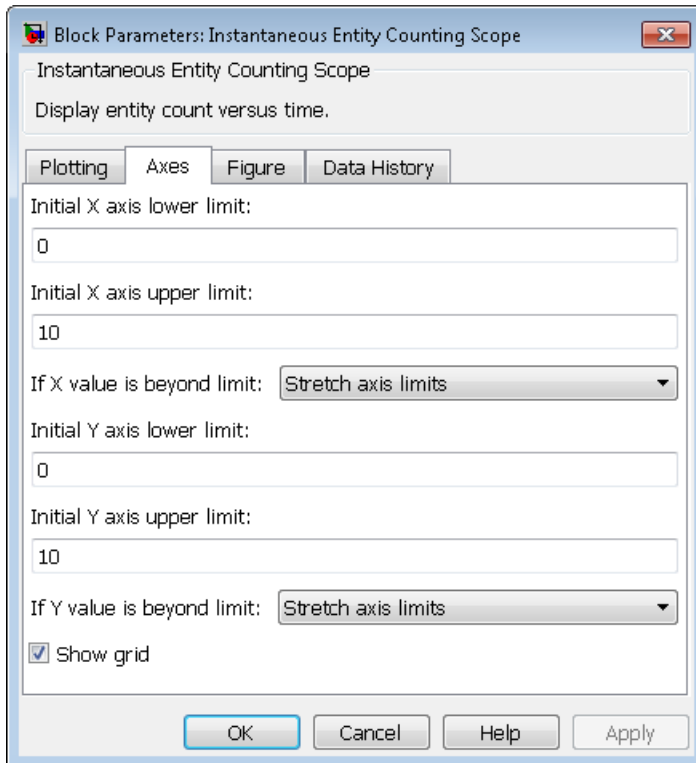
Enable entity OUT port

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 "Vary Axis Limits Automatically".

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.

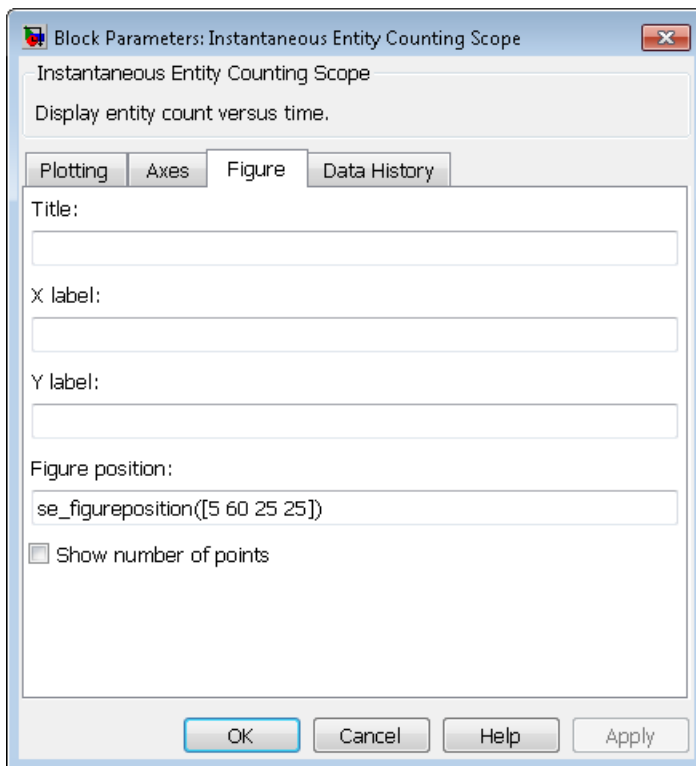
If Y value is beyond limit

Determines how the plot changes if one or more entity counts are not within the limits shown on the Y axis. For details, see “Vary Axis Limits Automatically”.

Show grid

Toggles the grid on and off.

Figure Tab



Title

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

Y label

Text that appears to the left of the vertical axis.

X label

Text that appears below the horizontal axis.

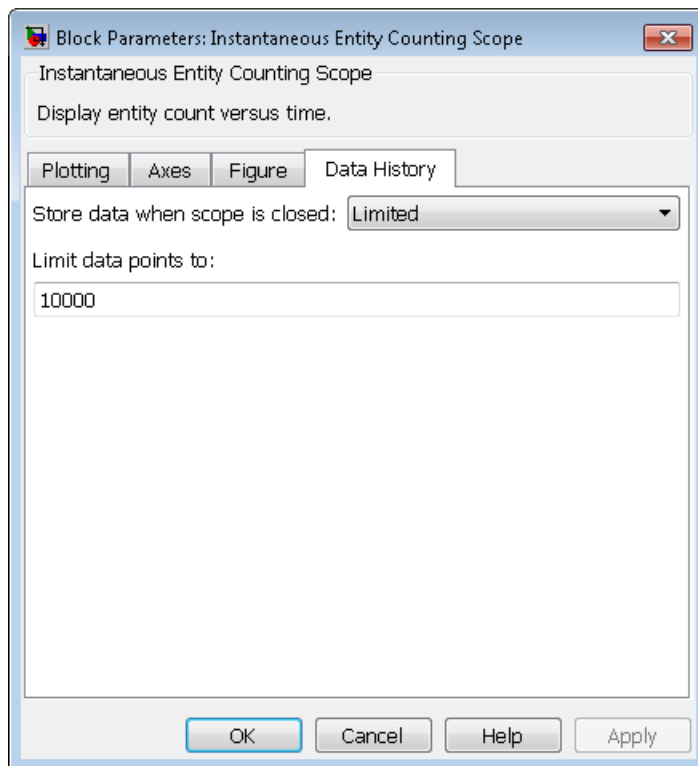
Position

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

Show number of entities

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

Data History Tab



Store data when scope is closed

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

Limit data points to

The number of data points the block caches, using the most recent data. You see this field only if you set **Store data when scope is closed** to **Limited**.

Examples

- “Count Simultaneous Departures from a Server”
- “Synchronize Service Start Times with the Clock”

See Also

Entity Departure Counter, Instantaneous Event Counting Scope

“Choose and Configure Plotting Blocks”, “Count Entities”

Instantaneous Event Counting Scope

Plot event count versus time

Library

SimEvents Sinks

Description



This block creates a plot by counting events. The block restarts the count from 1 when the time changes. As a result, the count is cumulative for a given time instant but not cumulative across the entire simulation.

When the block has a **ts** input port and the input signal is an event-based signal, a stem with no marker represents the initial output of the signal.

Ports

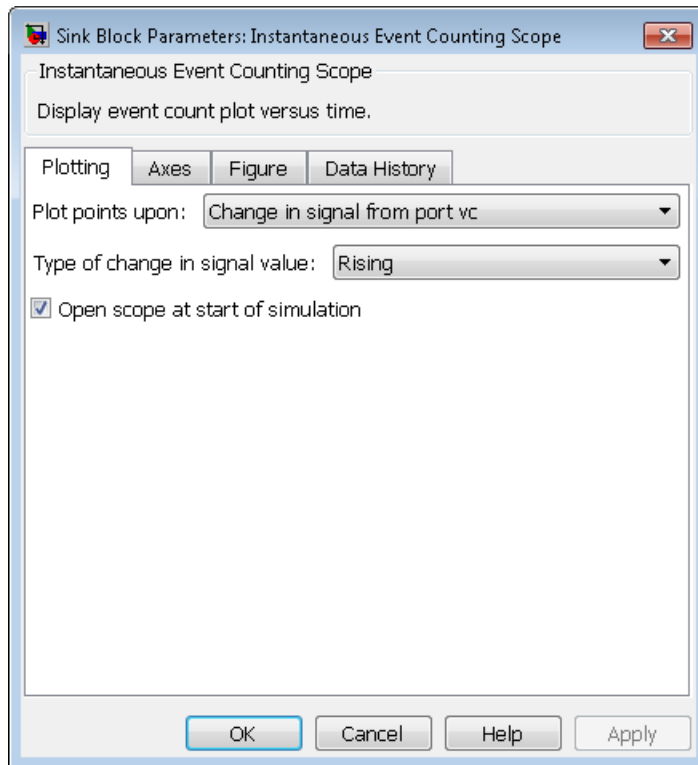
Signal Input Ports

Label	Description
ts	When this signal has an update, the counter increments. This signal must be an event-based signal. You see this port only if you set Plot points upon to Sample time hit from port ts .
tr	When this signal satisfies the specified trigger criteria, the counter increments. This signal must be an event-based signal. You see this port only if you set Plot points upon to Trigger from port tr .
vc	When this signal satisfies the specified value-change criteria, the counter increments. This signal must be an event-based signal. You see this port only if you set Plot points upon to Change in signal from port vc .
fcn	When this signal carries a function call, the counter increments. This signal must be an event-based function call. You see this port only if you set Plot points upon to Function call from port fcn .

Dialog Box

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

Plotting Tab



Plot points upon

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

Trigger type, Type of change in signal value

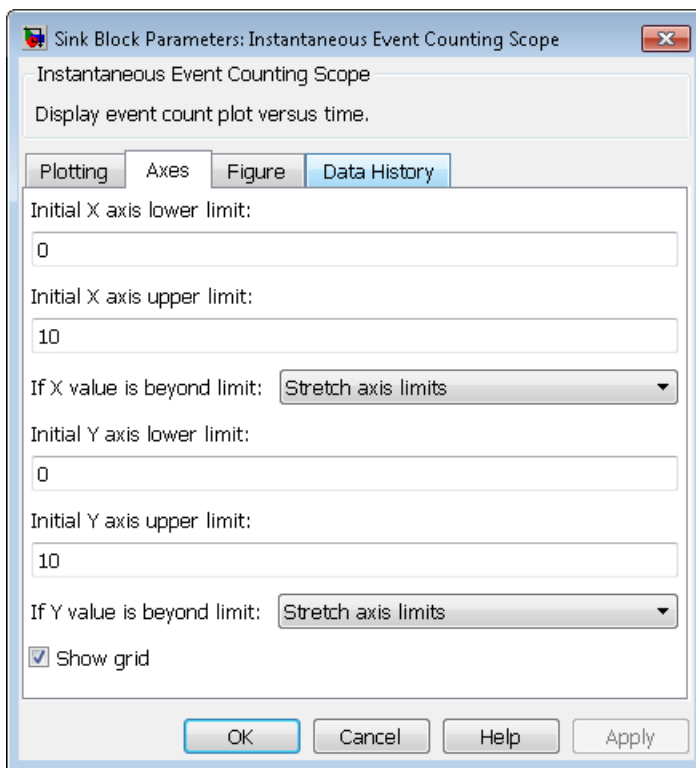
Trigger type determines whether rising, falling, or either type of trigger edge causes the block to increment its counter. You see this field only if you set **Plot points upon** to Trigger from port tr.

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

Open scope at start of simulation

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

Axes Tab



Initial X axis lower limit, Initial X axis upper limit

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

If X value is beyond limit

Determines how the plot changes if one or more X values are not within the limits shown on the X axis. For details, see “Vary Axis Limits Automatically”.

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.

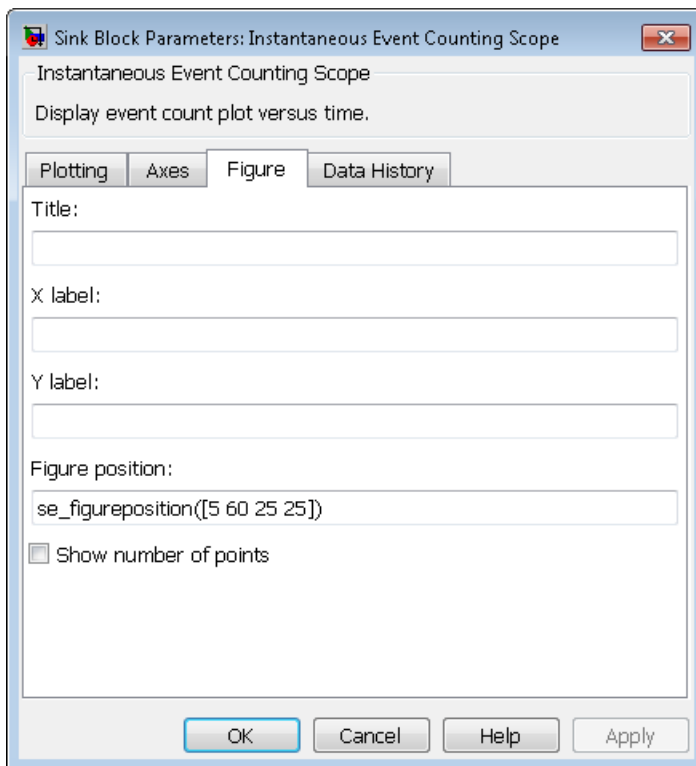
If Y value is beyond limit

Determines how the plot changes if one or more event counts are not within the limits shown on the Y axis. For details, see “Vary Axis Limits Automatically”.

Show grid

Toggles the grid on and off.

Figure Tab



Title

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

Y label

Text that appears to the left of the vertical axis.

X label

Text that appears below the horizontal axis.

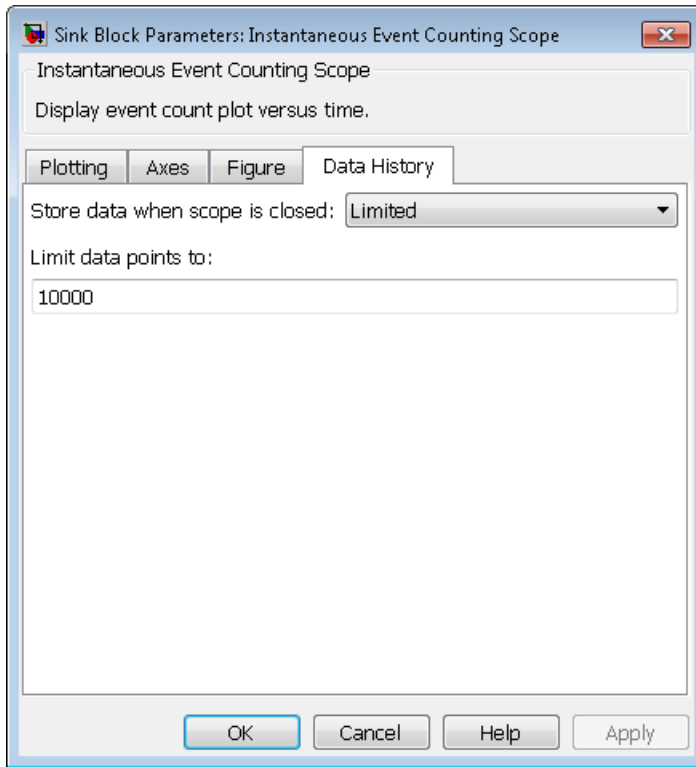
Position

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

Show number of points

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

Data History Tab



Store data when scope is closed

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

Limit data points to

The number of data points the block caches, using the most recent data. You see this field only if you set **Store data when scope is closed** to **Limited**.

Examples

See “Plot Event Counts to Check for Simultaneity”.

See Also

Signal Scope, Instantaneous Entity Counting Scope

“Choose and Configure Plotting Blocks”, “Observe Events”

LIFO Queue

Store entities in stack for undetermined length of time

Library

Queues



This block stores up to N entities simultaneously, where N is the **Capacity** parameter value. The block attempts to output an entity through the **OUT** port but retains the entity if the **OUT** port is blocked. If the block is storing multiple entities and no entity times out, then entities depart in a last-in, first-out (LIFO) fashion. If an entity in this block is scheduled to time out, then it might depart prematurely via the optional **TO** port. The length of time that an entity stays in this block cannot be determined in advance.

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

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities, which will be stored.

Entity Output Ports

Label	Description
OUT	Port for departing entities that do not time out while in this block.

Label	Description
TO	Port for entities that time out while in this block. You see this port only if you select Enable TO port for timed-out entities . This port must not be blocked when an entity attempts to depart here.

Signal Output Ports

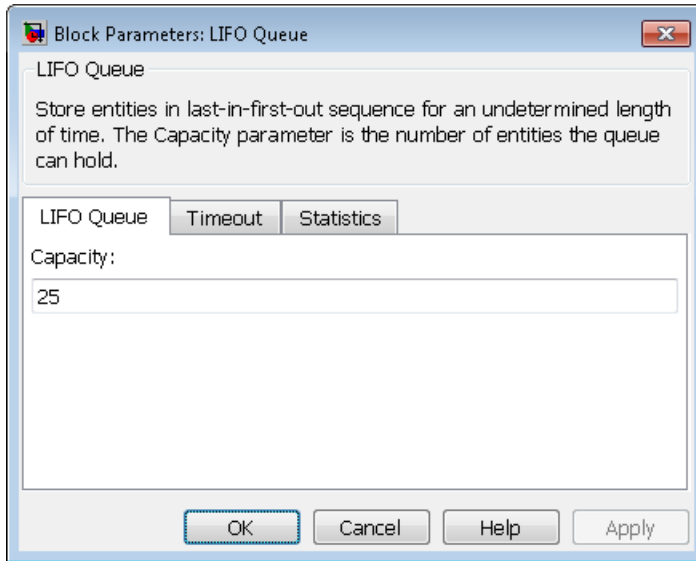
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block via the OUT port since the start of the simulation.	After entity departure via the OUT port	3
#n	Number of entities currently in the queue.	After entity arrival and after entity departure	2
w	Sample mean of the waiting times in this block for all entities that have departed via any port.	After entity departure	1
len	Average number of entities in the queue over time, that is, the time average of the #n signal.	After entity arrival and after entity departure.	1
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the TO port	3

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

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

Dialog Box

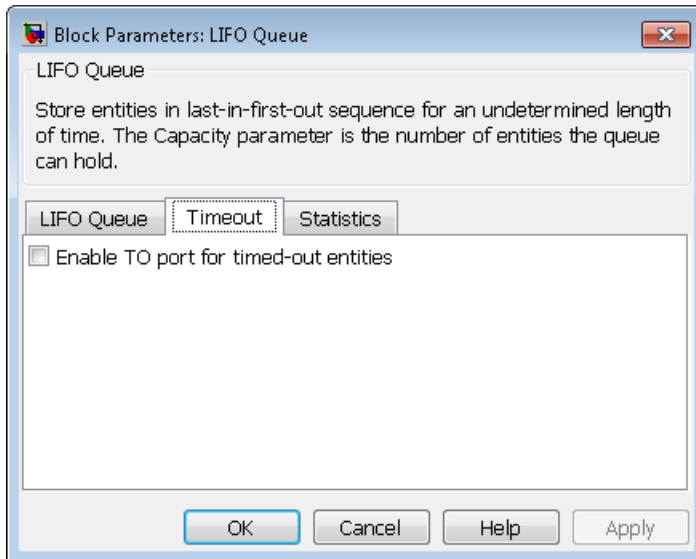
LIFO Queue Tab



Capacity

Determines how many entities the block can store at a time. The capacity must be a positive integer or Inf.

Timeout Tab

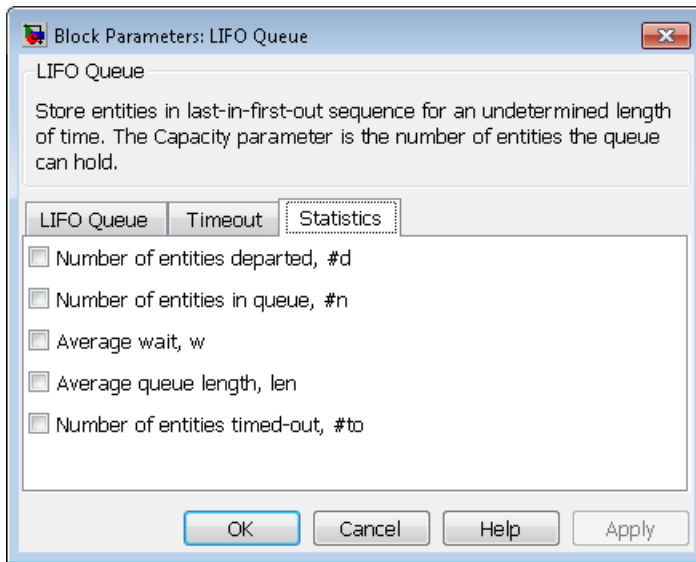


Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the Schedule Timeout block.

Statistics Tab

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



Number of entities departed

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

Number of entities in queue

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

Average wait

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

Average queue length

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

Number of entities timed out

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

Examples

See “LIFO Queue Waiting Time”.

See Also

FIFO Queue, Priority Queue

N-Server

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. For an example that uses the **TO** port of a queue block in the same way, see “Use Timeouts to Limit Entity Queueing Time”.

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”.

All entities that arrive do so via the **IN** port. The **IN** port is unavailable whenever this block contains N entities. In that case, the **IN** port becomes available when at least one of the N entities departs.

Ports

Entity Input Ports

Port Label	Description
IN	Port for arriving entities, which will be served.

Signal Input Ports

Port Label	Description
t	Service time, in seconds, for a newly arrived entity. This signal must be an event-based signal. You see this port only if you set Service time from to Signal port t .
pause	Port for input signal that disables all servers when the signal is positive. While the servers are disabled, any occupied servers retain their entities and the software pauses the remaining service time for each server. When the signal at the input port becomes nonpositive, each server resumes service. You see this port only if you select Allow service control and set Service change upon disabling to Pause .
complete	<p>Port for input signal that disables all servers when the signal is positive.</p> <p>When a positive signal enters the complete port, the software:</p> <ul style="list-style-type: none"> • Disables all servers. • Immediately completes service in all occupied servers. • Resets the remaining service time in all servers. <p>If no blockage exists at the entity output port of the N-Server block, entities immediately advance from occupied servers to downstream blocks. When the signal at the input port becomes nonpositive, normal behavior of the N-Server block resumes.</p> <p>You see this port only if you select Allow service control and set Service change upon disabling to Force complete.</p>

Entity Output Ports

Port Label	Description
OUT	Port for departing entities that have completed their service time and have not timed out while in this block.

Port Label	Description
TO	Port for entities that time out while in this block. You see this port only if you select Enable TO port for timed-out entities . This port must not be blocked when an entity attempts to depart here.

Signal Output Ports

Port Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block via the OUT port since the start of the simulation.	After entity departure via the OUT port	5
#n	Number of entities currently in the block, between 0 and N.	After entity arrival and after entity departure	4
pe	A value of 1 indicates that the block stores at least one entity that has tried and failed to depart. Such entities are pending entities. A value of 0 indicates that the block does not store any pending entities.	After the block stores an entity that has tried and failed to depart. In this case, the signal value is 1. After the departure of a pending entity. In this case, the signal value depends on whether any other pending entities remain in the block.	1
#pe	Number of pending entities in the block.	After the block stores an entity that has tried and failed to depart. After the departure of a pending entity.	3
w	Sample mean of the waiting times in this block for all entities that have departed via any port. An	After entity departure	2

Port Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
	entity's waiting time might exceed its service time if the OUT port is blocked when the entity completes service.		
util	Utilization of the N-server. If Number of servers is finite, util is the time average of the fraction of servers that are storing an entity. At time values when an entity arrives or departs, util equals $1/N$ times the time average of the #n signal. If Number of servers is infinite, then util is always zero.	Performance considerations cause the block to update the signal only after each arrival or departure of an entity.	2
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the TO port	5
so	Occupancy status of each server in the N-Server block. The so port outputs a vector of values. If a server is unoccupied, the value of the corresponding vector element is 0 . If a server is occupied, the vector element has a value of 1 .	After entity arrival and after entity departure	6

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

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

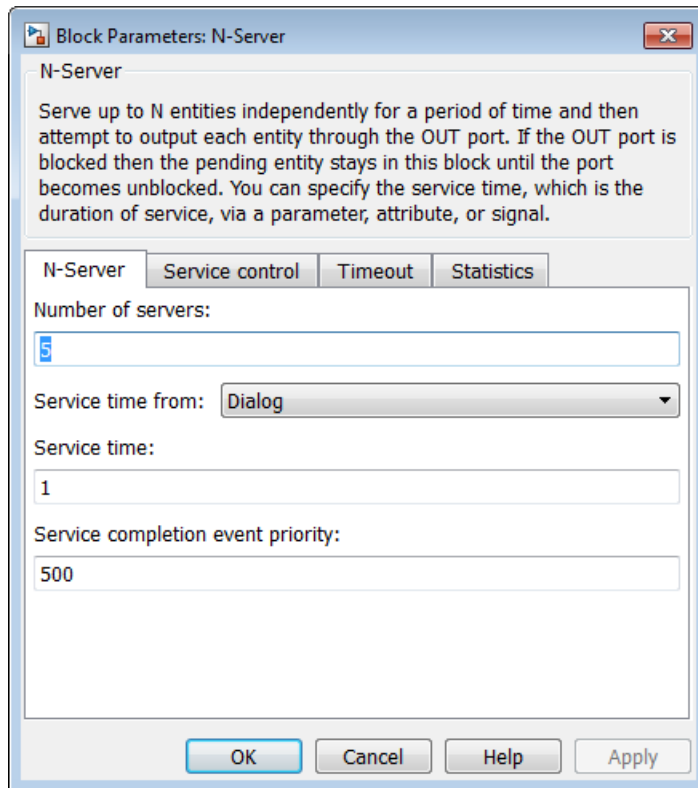
A more precise definition of the utilization signal **util** at an update time $T>0$ is

$$\frac{1}{T} \sum_k \left(\frac{(\#n)_k}{N} \right) \cdot \text{length}(I_k)$$

where I_k is the k th time interval between successive pairs of times that **util** is updated and $(\#n)_k$ is the number of entities the N-Server block is storing during the open interval I_k . If an update of **util** occurs at $T=0$, the value is $\#n/N$.

Dialog Box

N-Server Tab



Number of servers

The number of servers the block represents, N.

Service time from

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

Service time

The service time, in seconds, for all entities. You see this field only if you set **Service time from** to **Dialog**.

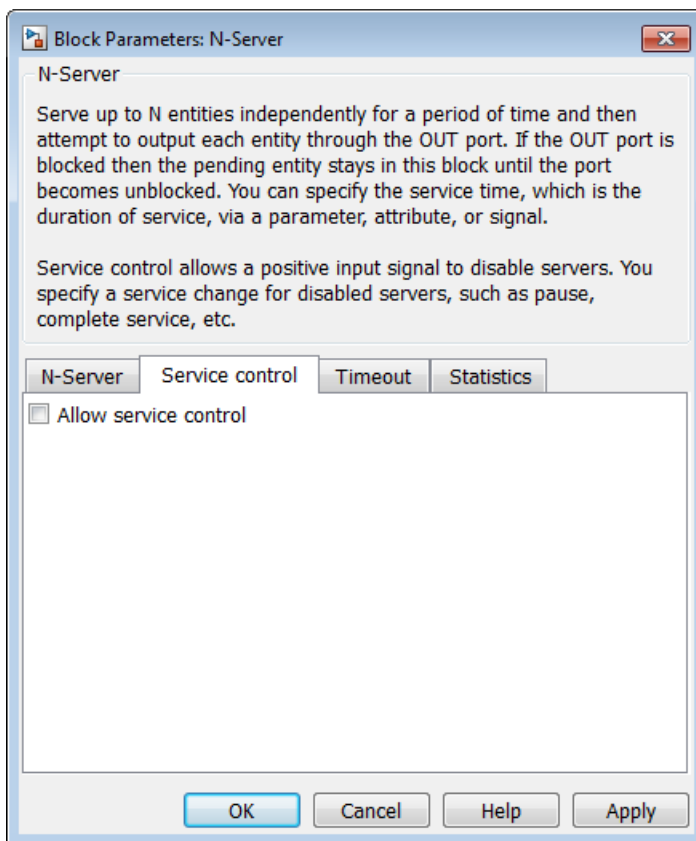
Attribute name

The name of the attribute whose value the block uses as the service time for an entity. You see this field only if you set **Service time from** to **Attribute**.

Service completion event priority

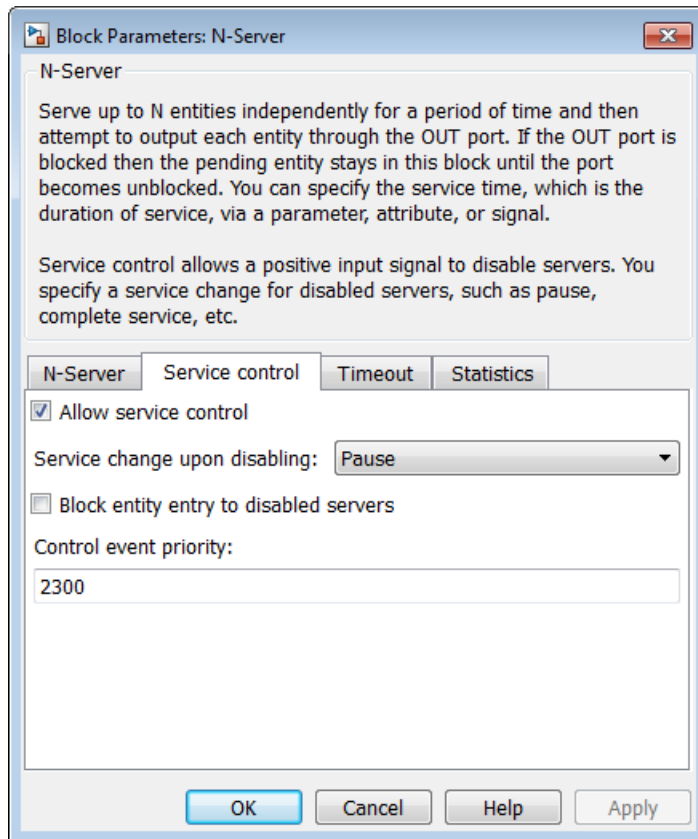
The priority of the service completion event, relative to other simultaneous events in the simulation.

Service control Tab



Allow service control

Adds an input signal port to the block. When you input a positive signal to this added signal port, the software disables servers in the block and applies a service change. You specify the service change action using an option that becomes visible when you select **Allow service control**.



Service change upon disabling

Specifies the service change action that the software applies to disabled servers. You see this option only if you select **Allow service control**.

By default, **Service change upon disabling** is set to **Pause**. When **Pause** is selected, the signal input port added by **Allow service control** is labeled **pause**.

When you input a positive signal to the **pause** port, the software disables all servers in the block. While this input signal remains positive, any occupied servers retain their entities and the software pauses the remaining service time for each server. When the signal at the input port becomes nonpositive, each server resumes service. For an example of using the **pause** port, see “Pause Service in a Conveyor Belt System”.

You can also set **Service change upon disabling** to **Force complete**. In this case, when you click **OK**, the label of the signal input port added by **Allow service control** changes to **complete**.

When a positive signal enters the **complete** port, the software:

- Disables all servers.
- Immediately completes service in all occupied servers.
- Resets the remaining service time in all servers.

If no blockage exists at the entity output port of the N-Server block, entities immediately advance from occupied servers to downstream blocks. When the signal at the input port becomes nonpositive, normal behavior of the N-Server block resumes.

For an example of using the **complete** port, see “Task Preemption in a Multitasking Processor”.

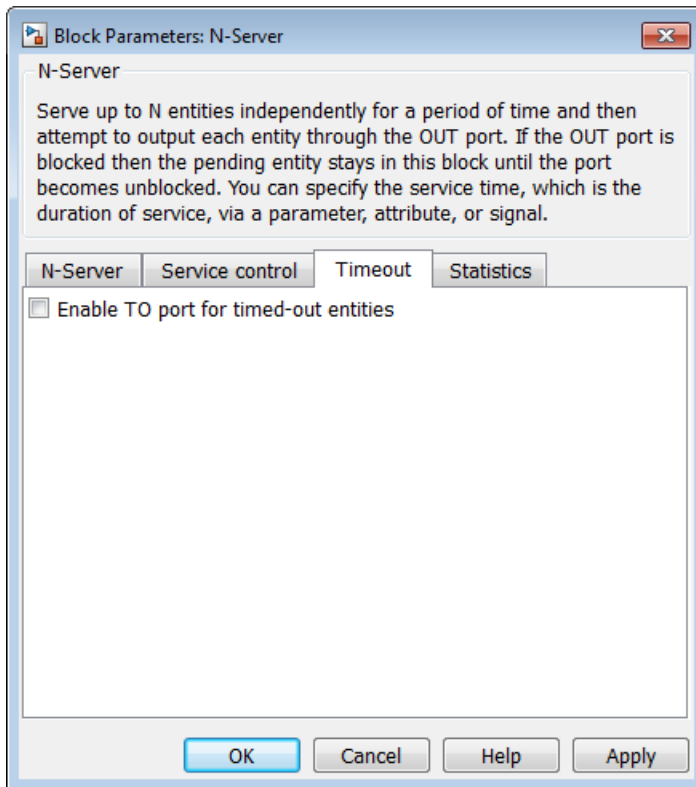
Block entity entry to disabled servers

Determines whether unoccupied servers can accept entities while the signal at the **pause** or **complete** port is positive.

Control event priority

The priority of the service control event, relative to other simultaneous events in the simulation.

Timeout Tab

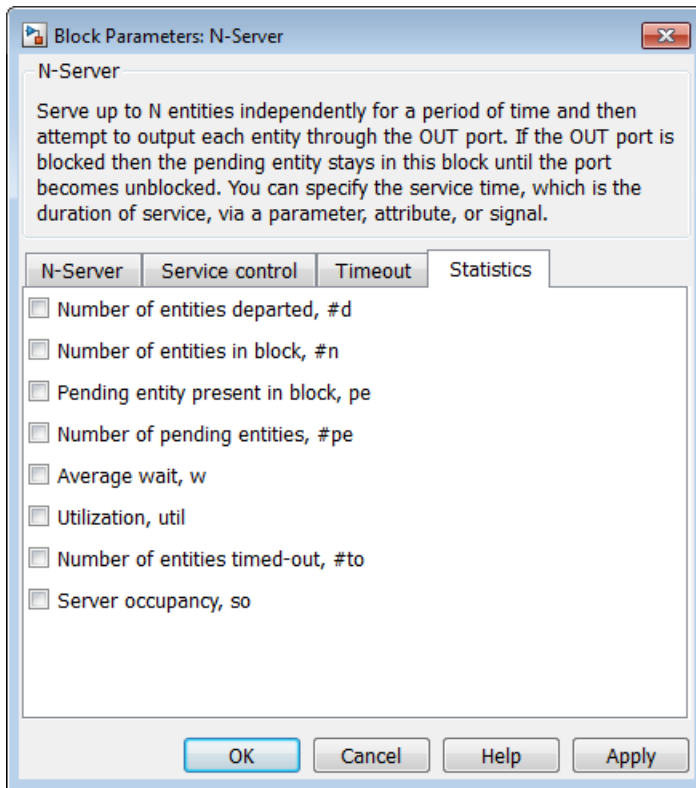


Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the Schedule Timeout block.

Statistics Tab

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



Number of entities departed

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

Number of entities in block

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

Pending entity present in block

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

Number of pending entities

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

Average wait

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

Utilization

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

Number of entities timed out

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

Server occupancy, so

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

Examples

See “Model an M/M/5 Queuing System”.

See Also

Single Server, Infinite Server

“Model Multiple Servers”

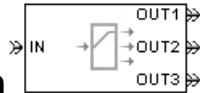
Output Switch

Select entity output port for departure

Library

Routing

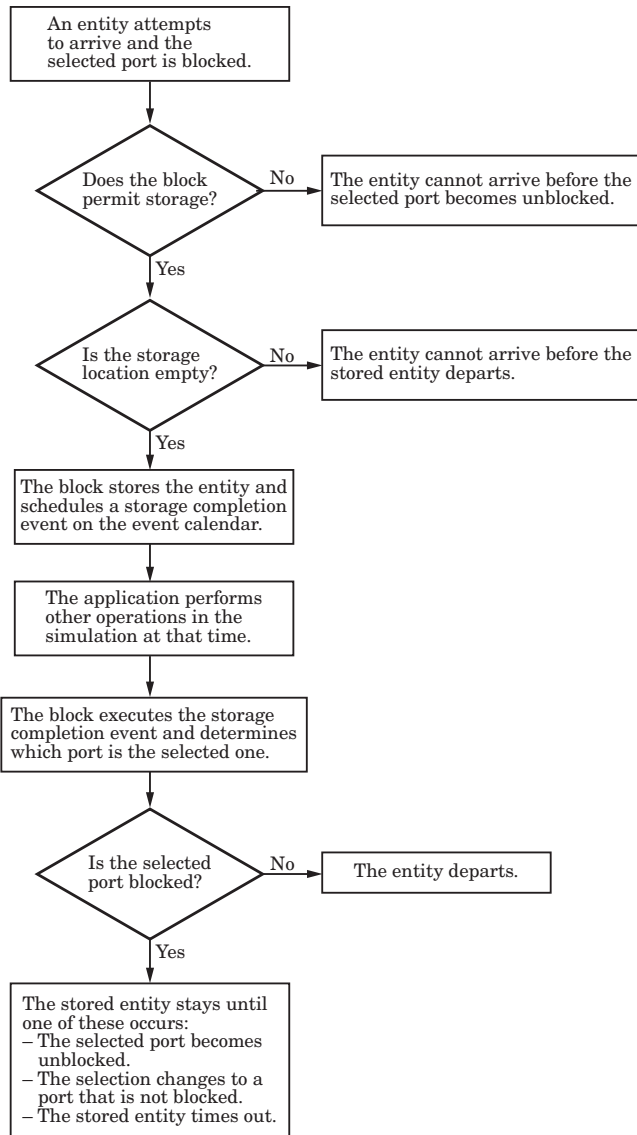
Description



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.



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 information about how you can handle such entities, see “Handle Timed-Out Entities”.

Switching Criteria

The **Switching criterion** parameter indicates how the block determines which entity output port is selected for departure at any given time. The values of the **Switching criterion** parameter are described in the table below.

Switching criterion Value	Description
Round robin	The first arriving entity in the simulation departs via the OUT1 port. Upon each subsequent arrival, the block selects the entity output port next to the last selected port. After exhausting all entity output ports, the block returns to the first one, OUT1 .
Equiprobable	At the beginning of the simulation and upon each departure, the block randomly chooses the entity output port through which the next arriving entity departs. All entity output ports are equally likely to be selected. The Initial seed parameter initializes the random number generation process.
First port that is not blocked	When an entity attempts to arrive, the block attempts to output the entity through OUT1 . If that port is blocked, then the block attempts to output the entity through OUT2 , and so on. If all entity output ports are blocked, then this block's IN port is unavailable and the entity cannot arrive.
From signal port p	Selecting this option creates an additional signal input port, labeled p . The signal at this port uses integer values between 1 and the Number of entity output ports parameter value to refer to entity output ports. The block monitors the p signal's value throughout the simulation and reacts to changes by selecting the corresponding entity output port.

Switching criterion Value	Description
From attribute	An arriving entity departs through the entity output port that corresponds to the value of an attribute of your choice. Name the attribute using the 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 “Use Signals To Route Entities”. For common problems and troubleshooting tips, see “Unexpected Use of Old Value of Signal”.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Signal Input Ports

Label	Description
p	Index of the entity output port through which an arriving entity departs. Values must be integers between 1 and Number of entity output ports . This signal must be an event-based signal. You see this port only if you set Switching criterion to From signal port p .

Entity Output Ports

Label	Description
OUT1, OUT2, OUT3, and so on	Entity ports through which an arriving entity departs, where the Switching criterion parameter determines which of multiple ports the entity departs through. The Number of entity output ports parameter determines how many of these entity output ports the block has.
TO	Port for entities that time out while in this block. You see this port only if you set Switching criterion to From signal port <i>p</i> , select Store entity before switching , and select Enable TO port for timed-out entities . This port must not be blocked when an entity attempts to depart here.

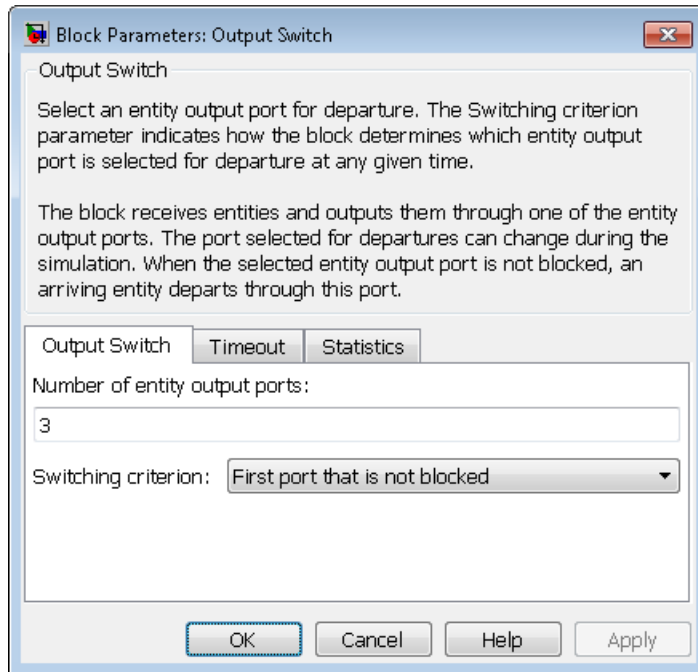
Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update
#d	Number of entities that have departed from this block without timing out, since the start of the simulation.	After entity departure via a port other than TO	3
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the TO port	2
pe	A value of 1 indicates that the block stores an entity that has tried and failed to depart. In that case, the entity is a pending entity. A value of 0 indicates that the block does not store any pending entities.	Sample time hit of 1 occurs after the block stores an entity that has tried and failed to depart. Sample time hit of 0 occurs after the departure of the pending entity via any port.	1
last	Index of the output port through which the last entity departed, excluding timed-out entities. Aside from the initial output, values of this signal are 1, 2, 3,..., Number of entity output ports .	After entity departure via a port other than TO	2

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

Dialog Box

Output Switch Tab



Number of entity output ports

Determines how many entity output ports the block has.

Switching criterion

The rule that determines which entity output port an arriving entity departs through.

Initial seed

A nonnegative integer that initializes the random number generator used to select an entity output port. You see this field only if you set **Switching criterion** to Equiprobable.

Specify initial port selection

Select this option to indicate the initially selected entity output port. For details, see “Specify an Initial Port Selection”. You see this field only if you set **Switching criterion** to From signal port *p*.

Initial port selection

The entity output port that the block selects when the simulation begins. The value must be an integer between 1 and **Number of entity output ports**. The block uses **Initial port selection** instead of the *p* signal's value until the signal has its first sample time hit. You see this field only if you set **Switching criterion** to From signal port *p* and select **Specify initial port selection**.

Store entity before switching

If you select this option, the block can store one entity at a time. Furthermore, the block decouples its arrival and departure processing to give other blocks in the simulation an opportunity to update the *p* signal if appropriate. If you do not select this option, the block processes an arrival and departure as an atomic operation and assumes that the *p* signal is already up to date at the given time. For details, see “Use the Storage Option to Prevent Latency Problems”. You see this field only if you set **Switching criterion** to From signal port *p*.

Resolve simultaneous signal updates according to event priority

Select this option to prioritize the port-selection event explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority SYS1 on the event calendar. For details, see “Resolve Simultaneous Signal Updates”. You see this field only if you set **Switching criterion** to From signal port *p* and do not select **Store entity before switching**.

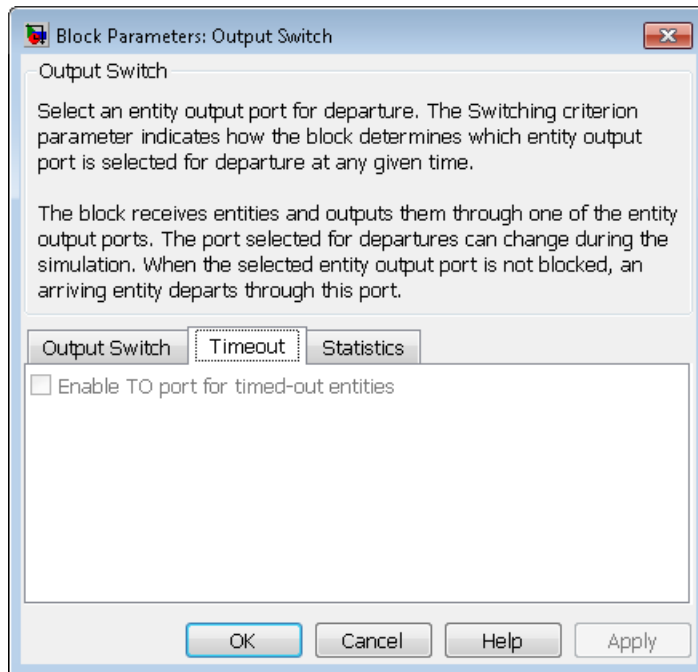
Event priority

The priority of the port-selection event, relative to other simultaneous events in the simulation. For details, see “Specify Event Priorities to Resolve Simultaneous Signal Updates”. You see this field 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. You see this field only if you set **Switching criterion** to From attribute.

Timeout Tab

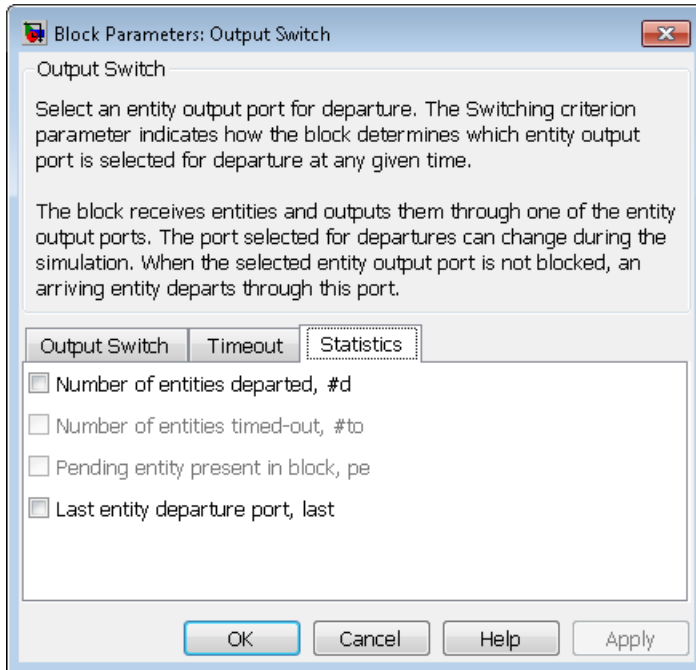


Enable TO port for timed-out entities

This option is available only if you set **Switching criterion** to **From signal port p**, and then select **Store entity before switching** on the **Output Switch** tab of the dialog box. This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the description of the **If entity has no destination when timeout occurs** parameter on the Schedule Timeout block reference page.

Statistics Tab

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



Number of entities departed

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

Number of entities timed out

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

Pending entity present in block, **pe**

Allows you to use the signal output port labeled **pe**. You can select this check box only if you set **Switching criterion** to **From signal port p**, and then select **Store entity before switching** on the **Output Switch** tab of the dialog box.

Last entity departure port

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

Examples

- “Select the First Available Server”

- “Use an Attribute to Select an Output Port”
- “Model a Packet Switch”
- “Queue Selection Using a Switch”

See Also

Input Switch

“Select Departure Path Using Output Switch”, “Use Signals To Route Entities”

Path Combiner

Merge entity paths

Library

Routing



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 “Simultaneous Events”. For an example, see the “No blockage” case in “Connect Multiple Queues to the Output Switch” 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.

Multiple instances of entities of the same type, but with different attributes, can arrive at the Path Combiner block. In these situations, the compiled entity type displays the union type.

Input Port Precedence

The **Input port precedence** parameter indicates how the block determines which entity input port to notify first, whenever the entity output port changes its status from blocked to unblocked. The first notified port is the first port to become available to an arriving entity. Choices for the **Input port precedence** parameter are described in the following table. For an example illustrating when this parameter is significant, see “Combine Entity Paths”.

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 signal must be an event-based signal. You see this port only if you set Input port precedence to From signal port p.

Entity Output Ports

Label	Description
OUT	Port for departing entities.

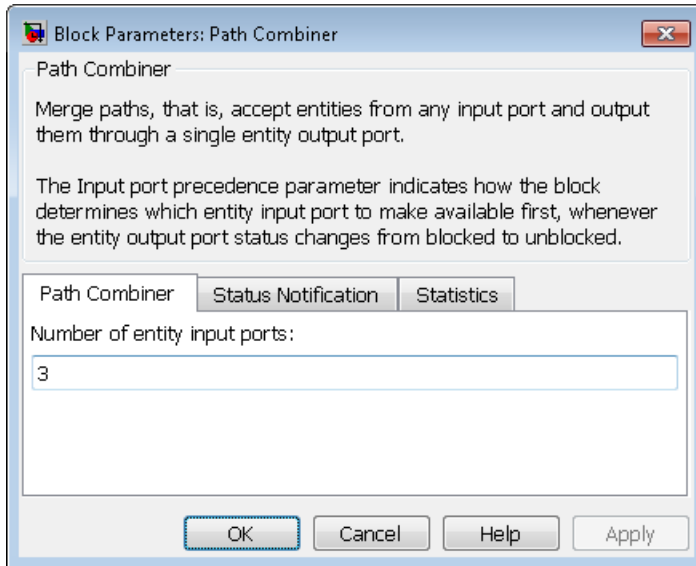
Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	2
last	Index of the input port through which the last entity arrived. The initial value is 0. After an entity has arrived and departed, values are 1, 2, 3,..., Number of entity input ports .	After entity departure	1

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

Dialog Box

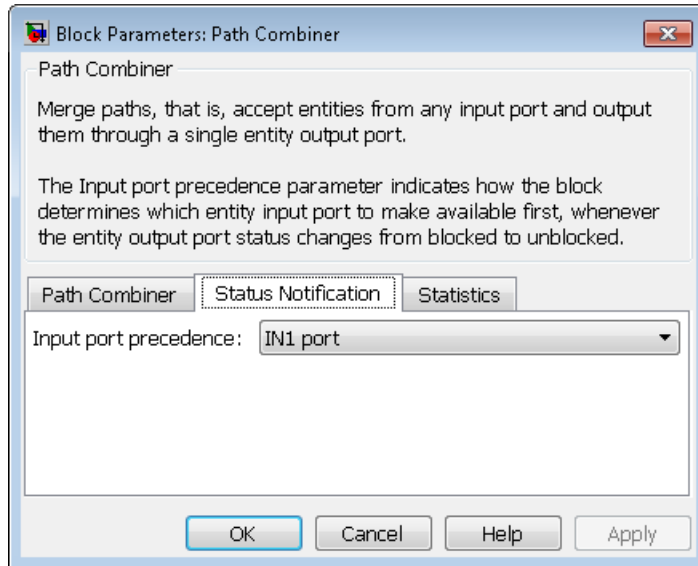
Path Combiner Tab



Number of entity input ports

Determines how many entity input ports the block has.

Status Notification Tab



Input port precedence

Determines which entity input port the block makes available first, upon an event that changes the entity output port from blocked to unblocked.

Initial seed

A nonnegative integer that initializes the random number generator used to select an entity input port for first notification about status changes. You see this field only if you set **Input port precedence** to Equiprobable.

Resolve simultaneous signal updates according to event priority

Select this option to prioritize the event that updates the port precedence explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority SYS1 on the event calendar. For details, see “Resolve Simultaneous Signal Updates”. You see this field only if you set **Switching criterion** to From signal port p.

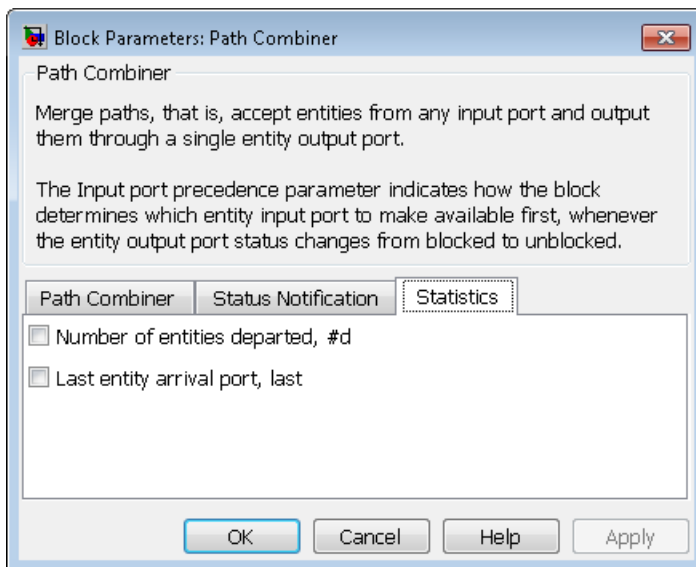
Event priority

The priority of the event that updates the port precedence, relative to other simultaneous events in the simulation. For details, see “Specify Event Priorities to Resolve Simultaneous Signal Updates”. You see this field only if you set **Switching**

criterion to From signal port **p** and select **Resolve simultaneous signal updates according to event priority**.

Statistics Tab

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



Number of entities departed

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

Last entity arrival port

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

Examples

- “Combine Entity Paths”
- “Model a Packet Switch”

- “Treat First Entity as Special Case”

See Also

Input Switch, Output Switch

“Combine Entity Paths”

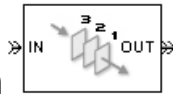
Priority Queue

Store entities in sorted sequence for undetermined length of time

Library

Queues

Description



This block stores up to N entities simultaneously in a sorted sequence, where N is the **Capacity** parameter value. The queue sorts entities according to the values of an attribute, in either ascending or descending order. Use the **Sorting attribute name** and **Sorting direction** parameters to determine the sorting behavior. The block accepts real numbers, Inf , and $-Inf$ as valid values of the sorting attribute.

The block attempts to output an entity through the **OUT** port but retains the entity if the **OUT** port is blocked. If an entity in this block is scheduled to time out, then it might depart prematurely via the optional **TO** port. The length of time that an entity stays in this block cannot be determined in advance. The **IN** port is unavailable whenever this block stores exactly N entities. In this case, the queue is said to be full.

While you can view the value of the sorting attribute as an entity priority, this value has nothing to do with event priorities or block priorities.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities, which will be stored.

Entity Output Ports

Label	Description
OUT	Port for departing entities that do not time out while in this block.
TO	Port for entities that time out while in this block. You see this port only if you select Enable TO port for timed-out entities . This port must not be blocked when an entity attempts to depart here.

Signal Output Ports

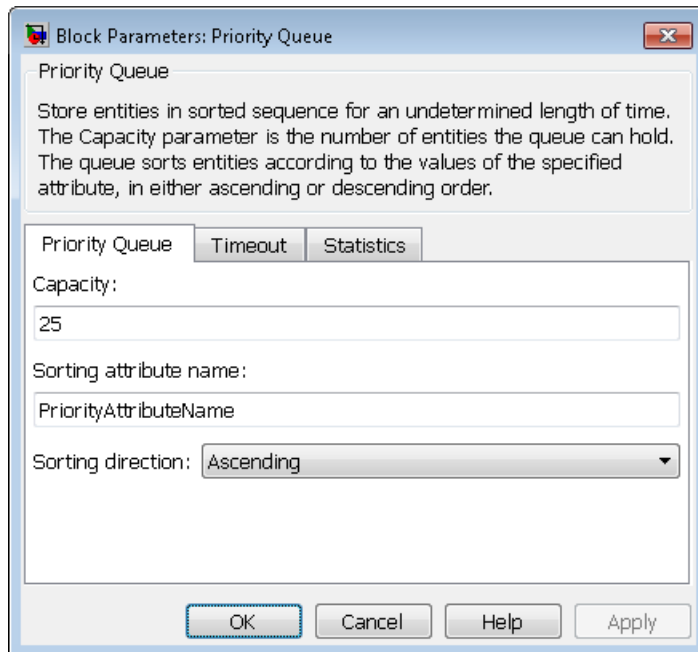
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block via the OUT port since the start of the simulation.	After entity departure via the OUT port	3
#n	Number of entities currently in the queue.	After entity arrival and after entity departure	2
w	Sample mean of the waiting times in this block for all entities that have departed via any port.	After entity departure	1
len	Average number of entities in the queue over time, that is, the time average of the #n signal.	After entity arrival and after entity departure.	1
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the TO port	3

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

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

Dialog Box

Priority Queue Tab



Capacity

Determines how many entities the block can store at a time. The capacity must be a positive integer or `Inf`.

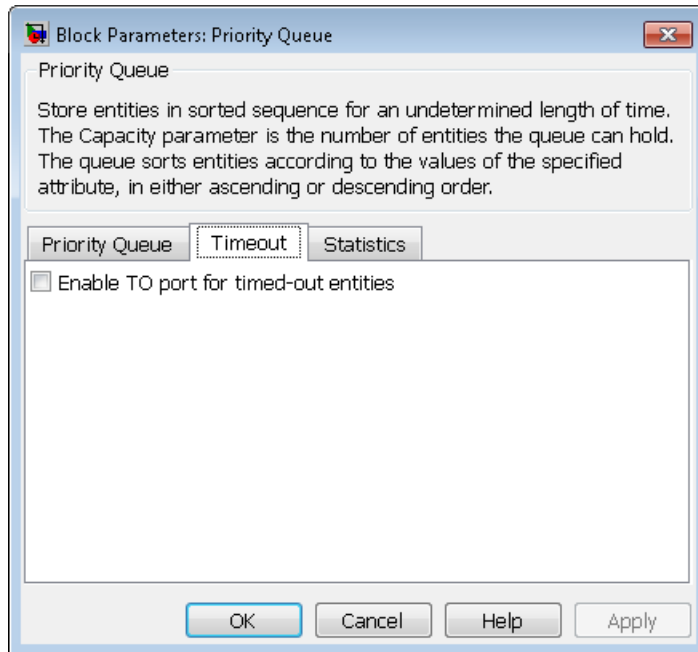
Sorting attribute name

The block uses this attribute to sort entities in the queue.

Sorting direction

Determines whether the entity at the head of the queue is the one with the smallest (**Ascending**) or largest (**Descending**) value of the attribute named above. Entities sharing the same attribute value are sorted in FIFO sequence.

Timeout Tab

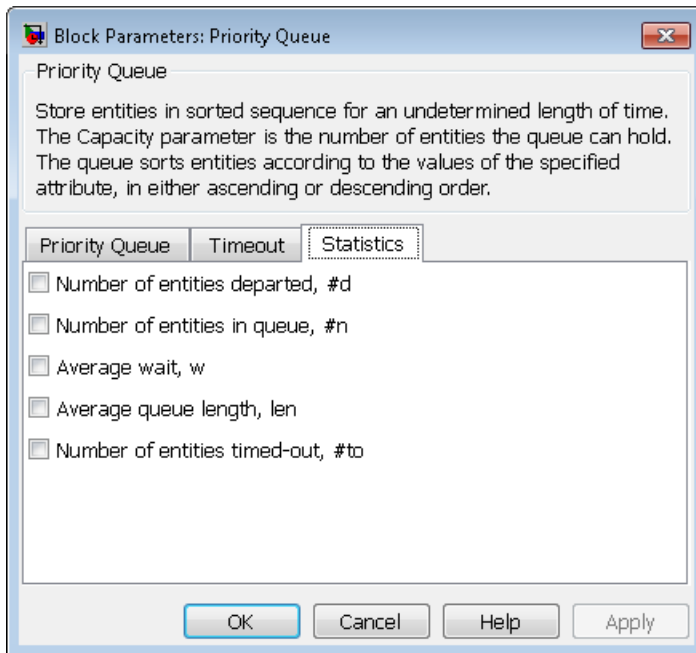


Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the Schedule Timeout block.

Statistics Tab

These parameters determine whether certain ports produce data throughout the simulation, produce data only when you stop or pause the simulation, or are omitted from the block. For descriptions of the affected ports, see the preceding table, "Signal Output Ports".



Number of entities departed

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

Number of entities in queue

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

Average wait

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

Average queue length

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

Number of entities timed out

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

Examples

- “Serve Preferred Customers First”

- “Preemption by High-Priority Entities”
- “Reroute Timed-Out Entities to Expedite Handling”

See Also

FIFO Queue, LIFO Queue, Single Server

“Sort by Priority”

Read Timer

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 “Measure Point-to-Point Delays”.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Entity Output Ports

Label	Description
OUT	Port for departing entities.

Signal Output Ports

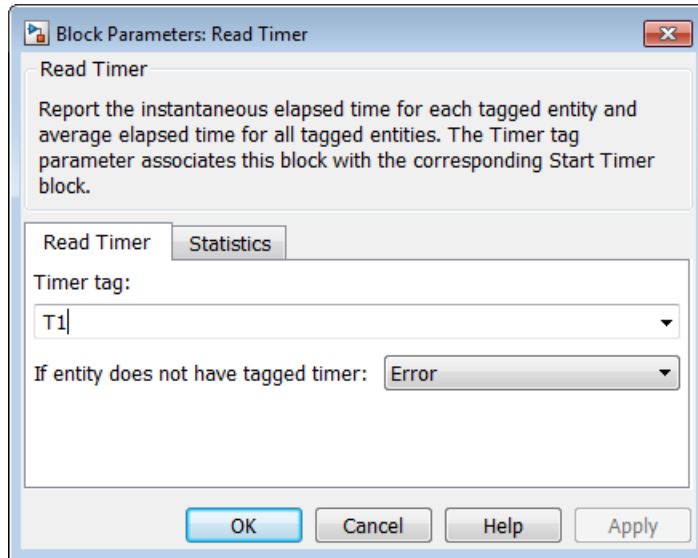
Label	Description	Time of Update When Statistic Is On	Order of Update
#d	Number of entities that have departed from this block since the start of the simulation.	After entity departure	3
#t	Total number of entities that have departed from this block and possessed a timer of the specified name.	After 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

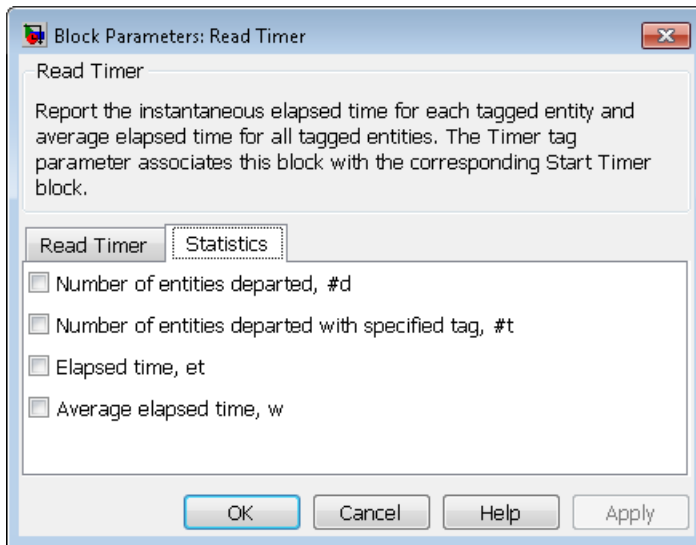
Name of the timer to read. This timer tag corresponds to the **Timer tag** parameter of a Start Timer block in the model.

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 the block produces data at signal output ports or omits those ports.



Number of entities departed

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

Number of entities departed with specified tag

Allows you to use the signal output port labeled **#t**. If you set **If entity does not have tagged timer** to **Ignore**, then the **#t** value might be less than the **#d** value.

Elapsed time

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

Average elapsed time

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

Examples

- “Basic Procedure for Using Timer Blocks”
- “Time Multiple Entity Paths with One Timer”
- “Restart a Timer from Zero”
- “Time Multiple Processes Independently”

See Also

Start Timer

“Measure Point-to-Point Delays”

Release Gate

Allow one pending entity to arrive when event occurs

Library

Gates

Description



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

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

The **Open gate upon** parameter determines the type of event that opens the gate:

- Sample time hits of a signal
- Edges in a trigger signal
- Changes in the numerical value of a signal
- Function calls

For more details, see “Open a Gate Instantaneously”.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Signal Input Ports

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

Entity Output Ports

Label	Description
OUT	Port for departing entities.

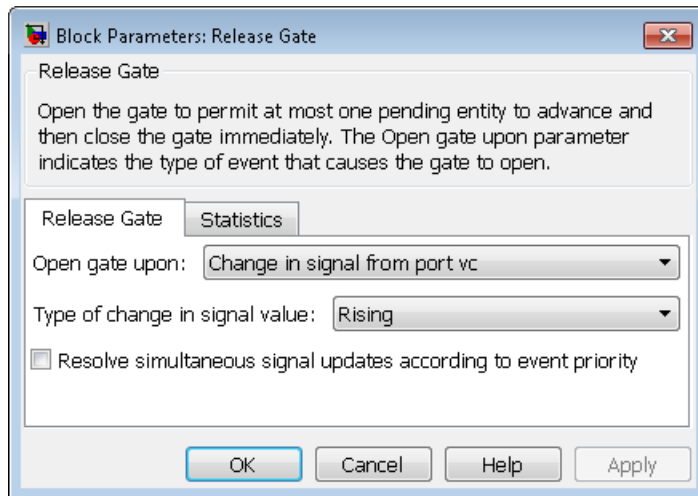
Signal Output Ports

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

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

Dialog Box

Release Gate Tab



Open gate upon

Determines the type of event that causes the gate to open instantaneously.

Trigger type, Type of change in signal value

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

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

Resolve simultaneous signal updates according to event priority

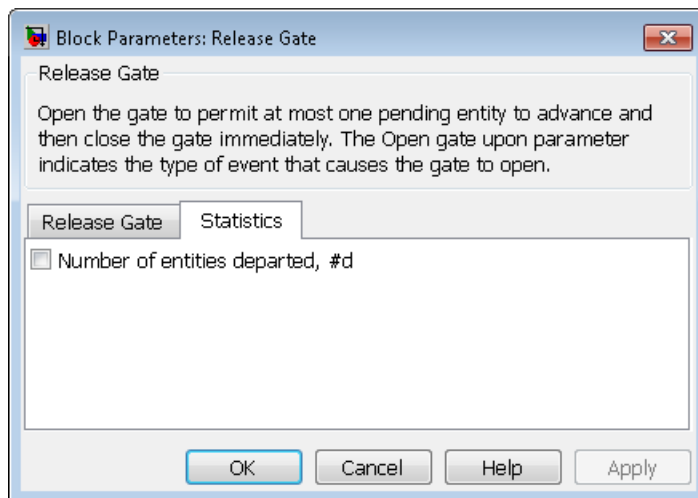
Select this option to prioritize the gate-opening event explicitly, relative to other simultaneous events in the simulation. If you do not select this option, the event has priority SYS1 on the event calendar. For details, see “Resolve Simultaneous Signal Updates”.

Event priority

The priority of the gate-opening event, relative to other simultaneous events in the simulation. For details, see “Specify Event Priorities to Resolve Simultaneous Signal Updates”. You see this field only if you select **Resolve simultaneous signal updates according to event priority**.

Statistics Tab

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



Number of entities departed

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

Examples

- “Synchronize Service Start Times with the Clock”
- “Treat First Entity as Special Case”

See Also

Enabled Gate

“Role of Gates in SimEvents Models”

Replicate

Output copies of entity

Library

Routing



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 more details about timeout events, see “Role of Timeouts in SimEvents Models” and “Use Timeouts to Limit Entity Queueing Time”.

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.

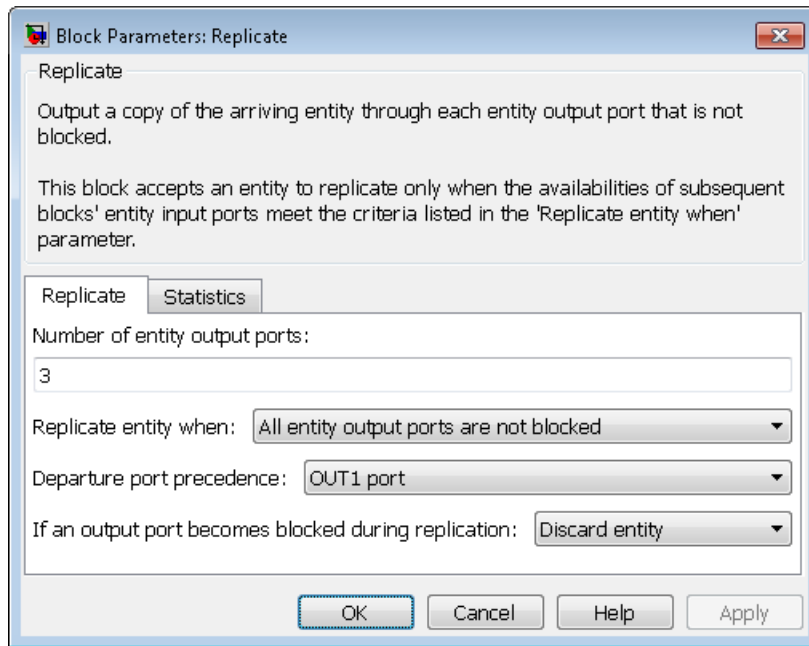
Signal Output Ports

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.

Replicate entity when

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

Departure port precedence

Determines the start of the sequence in which the block outputs the copies, each time the block replicates an entity.

Initial seed

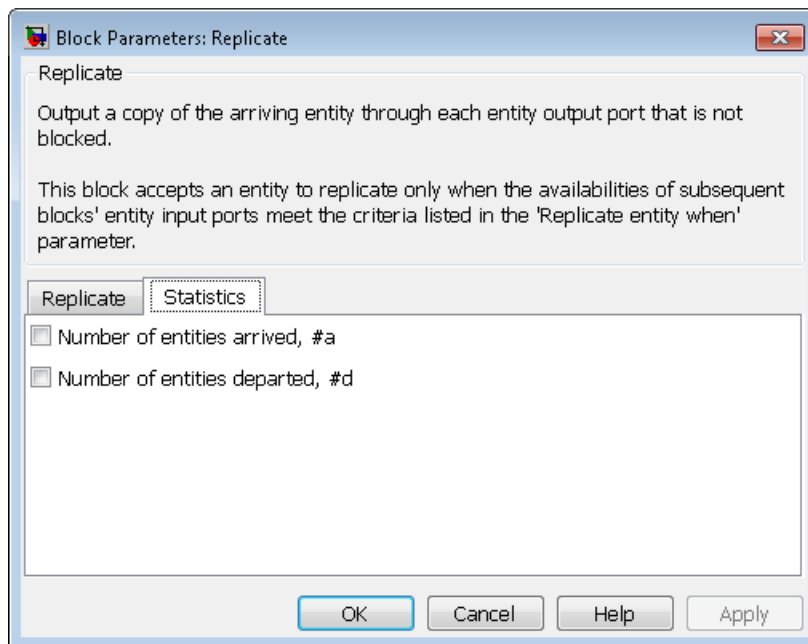
A nonnegative integer that initializes the random number generator used to determine the output sequence. You see this field only if you set **Departure port precedence** to Equiprobable.

If an output port becomes blocked during replication

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

Statistics Tab

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



Number of entities arrived

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

Number of entities departed

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

Examples

- “LIFO Queue Waiting Time”
- Replicating Entities example
- Communication Protocol Modeling in an Ethernet LAN example

See Also

Event-Based Entity Generator, Path Combiner

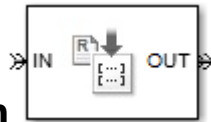
“Replicate Entities on Multiple Paths”

Resource Acquire

Acquire resource

Library

Entity Management



Description

This block accepts an entity that requests the use of resources, assigns resources to it, and then outputs it.

You can specify the resource types and amounts for the entity. The block stores the assigned resource with the entity, where each resource has a name and a value.

You can optionally specify a timeout that limits the maximum duration an entity waits for resources. You can also prioritize how resources are granted.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Entity Output Ports

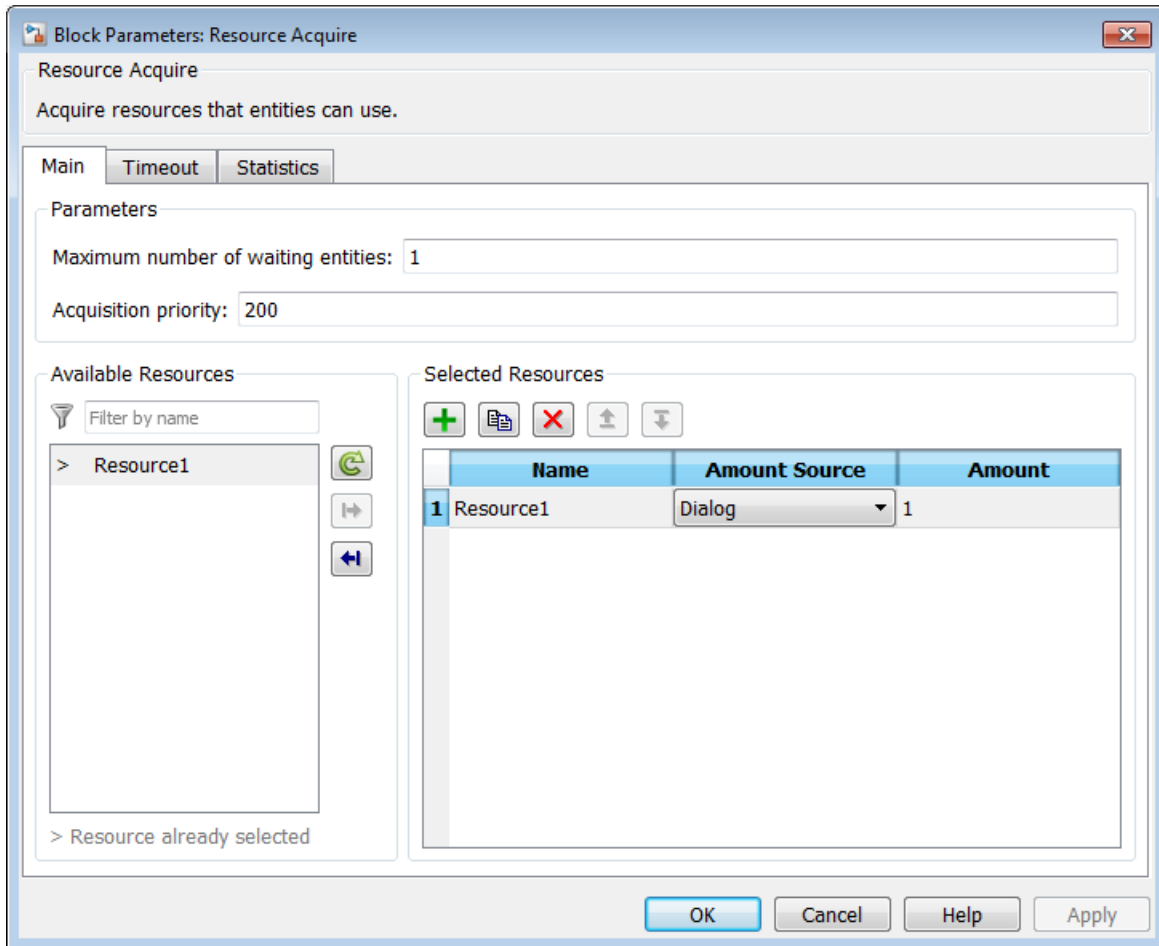
Label	Description
OUT	Port for departing entities that have acquired resources.

Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block since the start of the simulation or since the last reset.	After entity departure	3
#n	Number of entities currently in the block, between 0 and N .	After entity arrival and after entity departure	2
w	Sample mean of the waiting times in this block for all entities that have departed via any port. If the OUT port is blocked when the entity completes service, an entity waiting time exceeds its service time.	After entity departure	1
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the TO port	3

Dialog Box

Main Tab



Maximum number of waiting entities

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

Acquisition priority

Enter priority number for resource acquisition. This number prioritizes the Resource Acquire block that has higher priority in a model that has multiple Resource Acquire blocks. A smaller numeric value indicates higher priority.

Available Resources




Use the **Available Resources** controls to.

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

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

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

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

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

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

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






Selected Resources

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

Using these controls, you can:

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

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

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

Note: If you delete a row and apply the change, the deletion can affect signal output ports corresponding to other attributes. For example, if the block has a signal output port **A2** and you delete the attribute with a port marked **A1**, the block relabels **A2** as **A1**. Verify that any signal that connects to the relabeled port is still connected as you expect.

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

Timeout Tab

Enable TO port for timed-out entities

Select to use the signal output port labeled **TO**.

Statistics Tab

Number of entities departed, #d

Select to use the signal output port labeled **#d**.

Number of entities in block, #n

Select to use the signal output port labeled **#n**.

Average wait, w

Select to use the signal output port labeled **w**.

Number of entities timed out, #to

Select to use the signal output port labeled **#to**.

Examples

- Batch Production Process
- Kanban Production System
- Resource Allocation from Multiple Pools

See Also

Resource Pool, Resource Release

“Model with Resources”

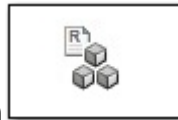
Resource Pool

Define resource

Library

Entity Management

Description



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

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

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

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

Ports

Resource Input Ports

Label	Description
amount	Port for arriving signals to specify the amount of the resource. You see this port only if you set Resource amount source to Signal port .

Signal Output Ports

Label	Description	Time of Update When Statistic is On
#u	Port to display the number of resources in use. You see this port only if you set	1

Label	Description	Time of Update When Statistic is On
	Resource amount source to Signal port.	
util	Port to display the average use of the resources. You see this port only if you select the Average utilization, util check box.	2

Dialog Box

Main Tab

Block Parameters: Resource Pool

Resource Pool
Define a resource that entities can acquire, use, and release.

Main **Statistics**

Resource name:
Resource1

Resource granularity: Discrete unit

Reusable upon release

Resource amount source: Dialog

Resource amount:
10

OK Cancel Help Apply

Resource name

Enter name of entity resource.

Resource granularity

Select granularity of resource use.

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

Reusable upon release

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

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

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

Resource amount source

Select resource amount source.

- **Dialog**

Selecting this option enables the **Resource amount** parameter.

- **Signal port**

The block derives the resource amount from the signal port.

Resource amount

Enter amount of resource.

Statistics Tab

Amount in use, #u

Select to use the signal output port labeled **#u**.

Average utilization, util

Select to use the signal output port labeled **util**.

Examples

- Batch Production Process

- Kanban Production System
- Resource Allocation from Multiple Pools

See Also

Resource Acquire, Resource Release

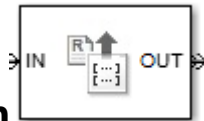
“Model with Resources”

Resource Release

Release resources that entities do not need

Library

Entity Management



Description

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

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Entity Output Ports

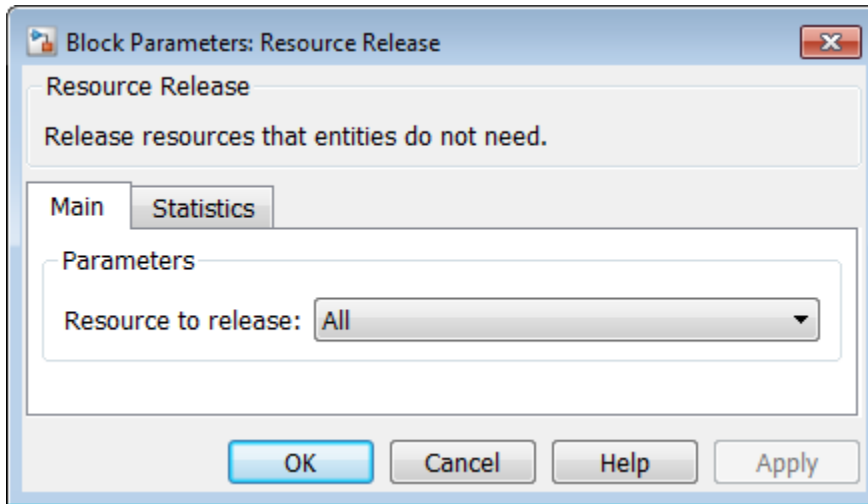
Label	Description
OUT	Port for departing entities that have released use of their resources.

Signal Output Ports

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

Dialog Box

Main Tab



Resource to Release

Select the resources to release.

- All

Release the use of all resources for a passing entity.

- Selected

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

Available Resources

Use the **Available Resources** controls to:




- Select the resources from the resources defined in all the Resource Pool blocks in the model

- Add the resources to the Selected Resources table, where you can modify them.

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

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

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

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

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

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

Selected Resources






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

Using these controls, you can:

- Add a resource manually.

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

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

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

Note: If you delete a row and apply the change, the deletion can affect signal output ports corresponding to other attributes. For example, if the block has a signal output port **A2** and you delete the attribute with a port marked **A1**, the block relabels **A2** as **A1**. Verify that any signal that connects to the relabeled port is still connected as you expect.

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

Statistics Tab

Number of entities departed, #d

Select to use the signal output port labeled **#d**.

Examples

- Batch Production Process
- Kanban Production System
- Resource Allocation from Multiple Pools

See Also

Resource Acquire, Resource Pool

“Model with Resources”

Schedule Timeout

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”.

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 “Assign Event Priorities” and “Processing Sequence for Simultaneous Events”.

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 signal must be an event-based signal. You see this port only if you set Timeout interval from to Signal port ti .

Entity Output Ports

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

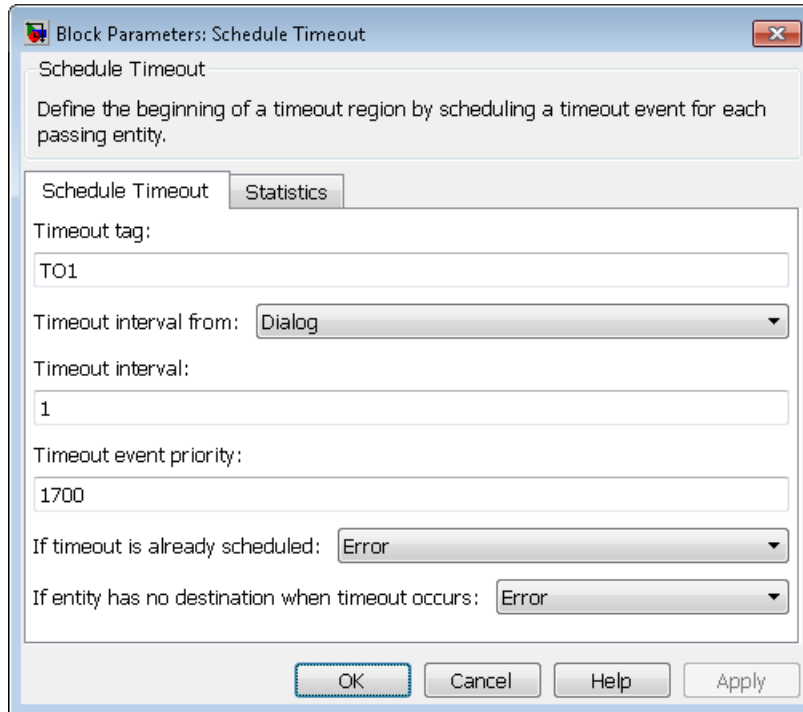
Signal Output Ports

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

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

Dialog Box

Schedule Timeout Tab



Timeout tag

Name of the timeout to associate with each entity. Enter a new timeout tag, or reschedule a previous timeout by choosing it in the drop-down list.

Timeout interval from

Determines whether the timeout interval is computed from a parameter in this dialog box, an input signal, or an attribute of the arriving entity.

Timeout interval

The length of time between an entity's arrival time and the scheduled timeout event for that entity. You see this field only if you set **Timeout interval from** to **Dialog**.

Attribute name

The name of the attribute whose value the block uses as the timeout interval for an entity. You see this field only if you set **Timeout interval from** to **Attribute**.

Timeout event priority

The priority of the timeout event, relative to other simultaneous events in the simulation.

If timeout is already scheduled

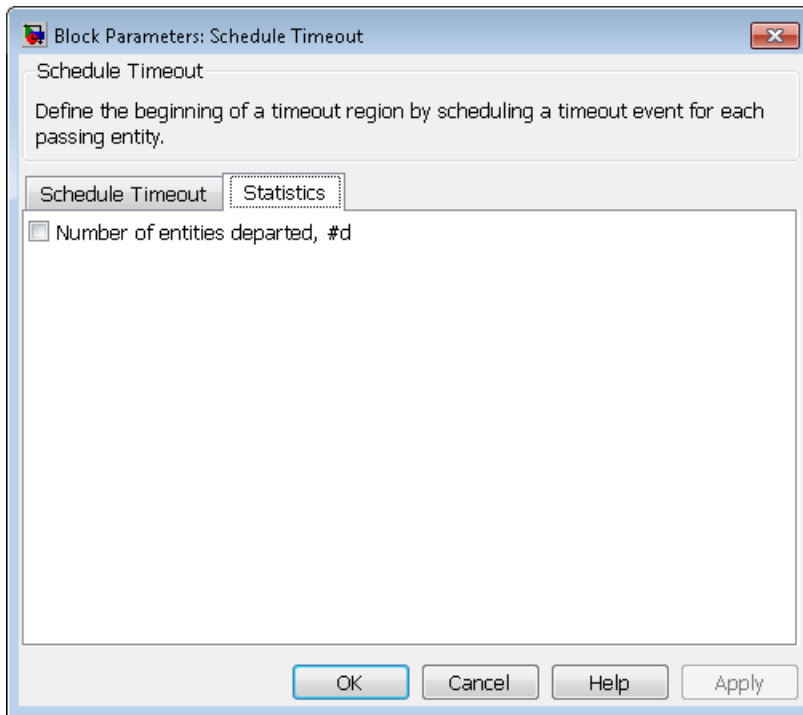
Behavior of the block if a timeout event with the specified timeout tag is already scheduled for the arriving entity.

If entity has no destination when timeout occurs

Behavior of the block if a timeout event occurs for an entity that resides in a block that has no visible **TO** entity output port.

Statistics Tab

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



Number of entities departed

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

Examples

- “Use Timeouts to Limit Entity Queueing Time”
- “Define Entity Timeout Paths”
- “Reroute Timed-Out Entities to Expedite Handling”
- “Limit the Time Until Service Completion”

See Also

“Workflow for Using Timeouts”

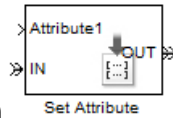
Cancel Timeout

Set Attribute

Assign data to entity

Library

Attributes

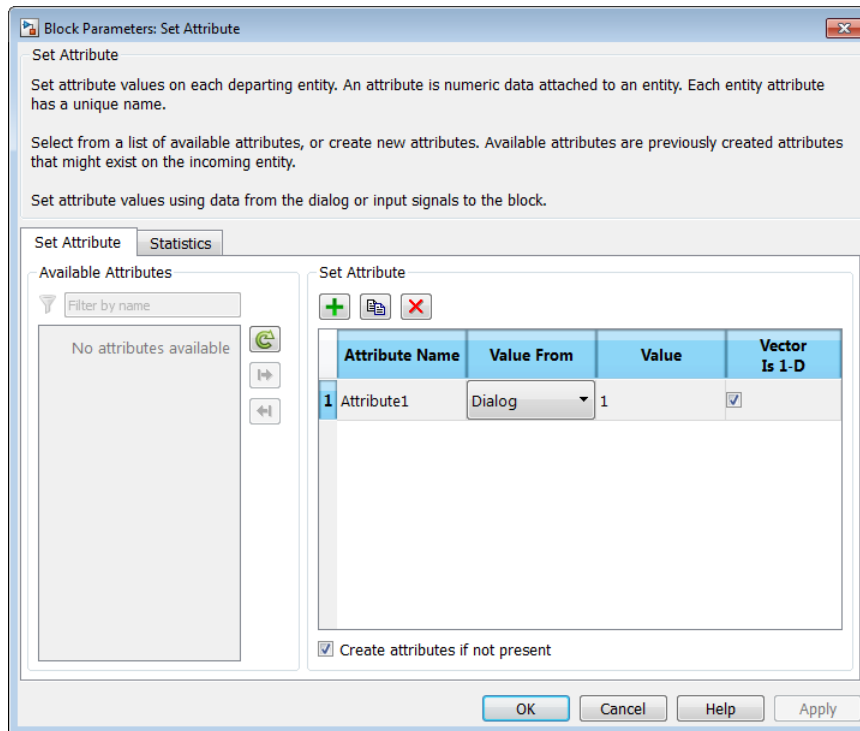


Description

This block accepts an entity, assigns data to it, and then outputs it. Assigned data is stored in entity attributes. Each attribute has a name and a value that you specify. You can specify up to 32 attributes in the block. To learn about the kind of data an attribute can store, see “Attribute Value Support”.

Dialog Box

Set Attribute Tab



Available Attributes




Use the **Available Attributes** controls to:

- Select the attributes from incoming entity paths that you want to access on the departing entity.
- Add the attributes to the **Set Attribute** table, where you can modify them.

The list displays all the attributes on all the incoming entities. (If the entity paths entering the Set Attribute block do not have any attributes, the **Available Attributes** list is empty).

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

Use the buttons in the **Available Attributes** section to help build the attributes table. The buttons perform these actions:

Button	Action
	Refresh the Available Attributes list. This action updates the list with any upstream model changes you make while the block dialog box is open.
	Add the selected attribute to the Set Attribute table.
	Move the selected attribute from the Set Attribute table to the Available Attributes list. Note: If the selected attribute is one you added manually, this button appears dimmed.

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

Message	Meaning
> Attribute already selected	You have already added the attribute to the Set Attribute table. You cannot add the attribute to the table again.
* Attribute may not be present	When multiple entity paths enter the block, all entities might not have the same attributes. Attributes that are not on all entering entities display an asterisk in the list, and this message appears. If you add such an attribute to the Set Attribute table, the behavior depends on how the Create attribute if not present check box is set.

Create attribute if not present

Check box that enables the block to define new attributes when an attribute in the table is not present in the current entity. If the check box is deselected, the simulation issues an error if an attribute named in the table does not already exist.

Select the check box if you want to:

- Set an attribute on each departing entity that previously existed on only certain incoming entity paths.
- Set a new attribute that you manually defined in the table on each departing entity.

Clear the check box if you want to:

- Protect against scenarios in which you add an attribute to the table from the **Available Attributes** list and later rename the attribute. When the check box is not selected, the renamed attribute causes an error because it no longer matches one present in the current entity.




Set Attribute

Use the controls under **Set Attribute** to build and manage the list of attributes to attach to each departing entity. Each attribute appears as a row in a table.

Using these controls, you can:

- Add an attribute manually to attach to the entity.
- Modify an attribute that you added to the table from the **Available Attributes** list to attach to the entity.

The buttons under **Set Attribute** perform these actions:

Button	Action	Notes
	Add a template attribute to the table.	Rename the attribute and specify its properties.
	Add a copy of the selected attribute to the table to use as the basis of a new attribute.	Rename the copy. Two attributes cannot have the same name.
	Remove the selected attribute from the Set Attribute table.	When you delete an attribute this way, no confirmation

Button	Action	Notes
		appears and you cannot undo the operation.

Note: If you delete a row and apply the change, the deletion can affect signal output ports corresponding to other attributes. For example, if the block has a signal output port **A2** and you delete the attribute with a port marked **A1**, the block relabels **A2** as **A1**. Verify that any signal that connects to the relabeled port is still connected as you expect.

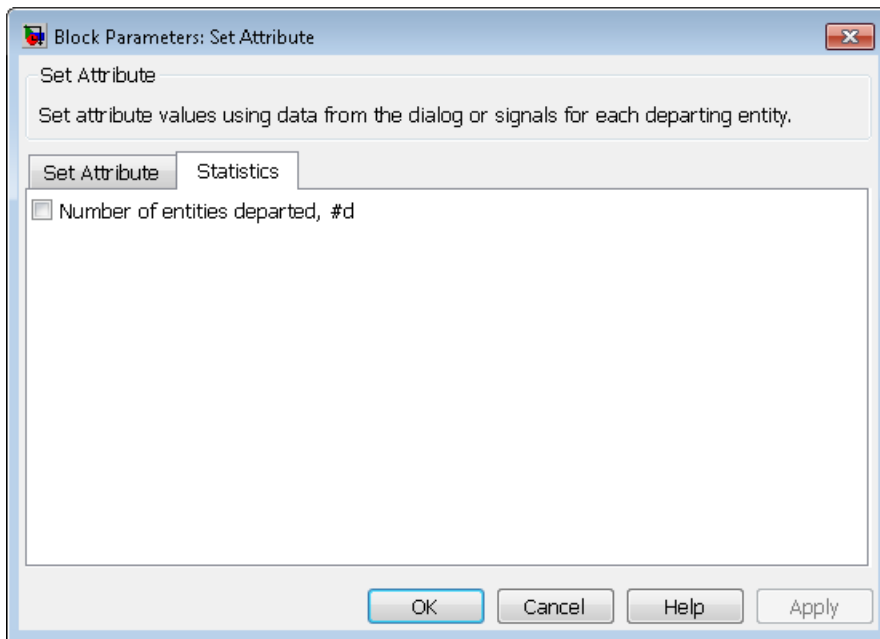
The table displays the attributes you added from the **Available Attributes** list or added manually. Use it to set these four attribute properties:

Property	Specify	Use
Attribute Name	The name of the attribute. Each attribute must have a unique name.	Double-click the existing name, and then type the new name.
Value From	Whether the data for the attribute value comes from the dialog box or a signal.	Select Dialog or Signal port . If you select Signal port , an input port with the name specified is added to the block after you apply your changes. When you connect a signal to the input port, the block assigns the value of the signal to the attribute during simulation.
Value	The value to assign to the attribute (when the attribute comes from the dialog box).	Double-click the value, and then type the value you want to assign.
Vector Is 1-D	Whether the block assigns the attribute as a vector of length N or as a multidimensional array when the Value evaluates to an N-element	Select the check box to assign the attribute as a vector of length N. Clear it to assign the attribute as a multidimensional array.

Property	Specify	Use
	row or column vector. This property applies only to values that come from the dialog box when the value evaluates to an N-element row or column vector.	

Statistics Tab

These parameters determine whether the block produces data at signal output ports or omits those ports. For descriptions of the data and ports, see [Signal Output Ports](#).



Number of entities departed

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

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities

Signal Input Ports

Label	Description
<i>Attribute name</i>	Data to assign to the attribute specified in each row of the table. The signal must be a fixed-size, event-based signal. You see this port only if you set Value From to Signal port . The default Name that corresponds to each row is Attributex , where x = 1, 2, 3, etc.

Entity Output Ports

Label	Description
OUT	Port for departing entities, with data assigned to them.

Signal Output Ports

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

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

Examples

- “Set Attributes”
- “Use an Attribute to Select an Output Port”
- “Round-Robin Approach to Choosing Inputs”
- “Reroute Timed-Out Entities to Expedite Handling”

See Also

Get Attribute

“Set Entity Attributes”

Signal Latch

Write input signal value to memory and read memory to output signal upon events

Library

Signal Management



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 Modeling Load Within a Dynamic Voltage Scaling Application example.

Ports

Signal Input Ports

Label	Description
wts	Signal whose updates cause write events. This signal must be an event-based signal. You see this port only if you set Write to memory upon to Sample time hit from port wts .
wtr	Trigger signal whose edges cause write events. This signal must be an event-based signal. You see this port only if you set Write to memory upon to Trigger from port wtr .
wvc	Signal whose numerical changes in value cause write events. This signal must be an event-based signal. You see this port only if you set Write to memory upon to Change in signal from port wvc .
wfcn	Function-call signal that causes write events. This signal must be an event-based function call. You see this port only if you set Write to memory upon to Function call from port wfcn .
rts	Signal whose updates cause read events. This signal must be an event-based signal. You see this port only if you set Read from memory upon to Sample time hit from port rts .
rtr	Trigger signal whose edges cause read events. This signal must be an event-based signal. You see this port only if you set Read from memory upon to Trigger from port rtr .
rvc	Signal whose numerical changes in value cause read events. This signal must be an event-based signal. You see this port only if you set Read from memory upon to Change in signal from port rvc .
rfcn	Function-call signal that causes read events. This signal must be an event-based function call. You see this port only if you set Read from memory upon to Function call from port rfcn .
in	Signal to be resampled and/or delayed. This signal must be an event-based signal.

Signal Output Ports

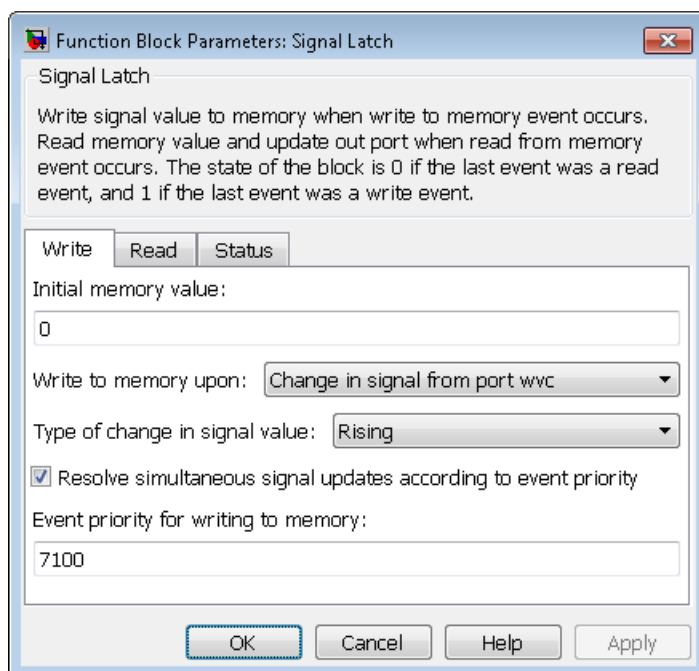
Label	Description	Time of Update When Statistic Is On	Order of Update	Initial Value
st	0 or 1, depending on whether the block more recently processed a read or write event.	Upon write events and upon read events	1	0
mem	The value of the block's internal memory when a write event occurs.	Upon write events	1	Value of Initial memory value parameter
out	The value of the block's internal memory when a read event occurs.	Upon read events	1	

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

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

Dialog Box

Write Tab



Initial memory value

The value in the block's internal memory before the first write event occurs.

Write to memory upon

The type of signal-based event or function call that causes a write event.

Trigger type, Type of change in signal value

Trigger type determines whether rising, falling, or either type of trigger edge causes a write event. You see this field only if you set **Write to memory upon** to **Trigger** from port wtr.

Type of change in signal value determines whether rising, falling, or either type of value change causes a write event. You see this field only if you set **Write to memory upon** to **Change in signal from port wvc**.

Resolve simultaneous signal updates according to event priority

Select this option to control the sequencing of the write event, relative to other simultaneous events in the simulation. If you do not select this option, the application executes the write event immediately upon detecting the signal-based event that causes it. For details, see “Resolve Simultaneous Signal Updates”.

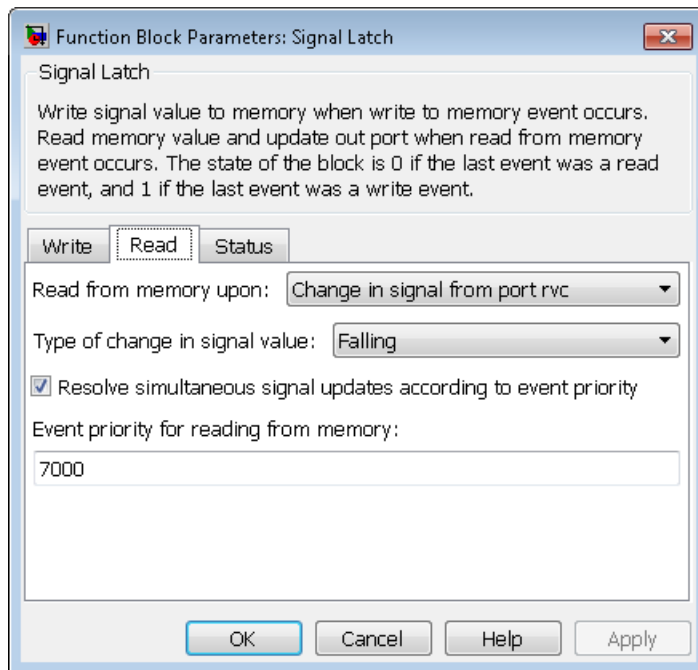
Event priority for writing to memory

The priority of the write event, relative to other simultaneous events in the simulation. For details, see “Specify Event Priorities to Resolve Simultaneous Signal Updates”.

Use of this parameter depends on the following:

- You see this field only if you select **Resolve simultaneous signal updates according to event priority** on this tab.
- If you select **Resolve simultaneous signal updates according to event priority** on both the write tab and the read tab, the software ignores **Event priority for reading from memory**. Instead, the simulation resolves simultaneous signal updates based on only the **Event priority for writing to memory** parameter. In this case, the software executes the write event before the read event.
- If you select the configuration parameter **Prevent duplicate events on multiport blocks and branched signals** in your model, the software uses the **Event priority for writing to memory** parameter to help Simulink to sort blocks in the model. In this case, the software no longer schedules an event that you can view on the SimEvents event calendar.

Read Tab



Read from memory upon

The type of signal-based event, function call, or internal write event that causes a read event.

Trigger type, Type of change in signal value

Trigger type determines whether rising, falling, or either type of trigger edge causes a read event. You see this field only if you set **Read from memory upon** to Trigger from port rtr.

Type of change in signal value determines whether rising, falling, or either type of value change causes a read event. You see this field only if you set **Read from memory upon** to Change in signal from port rvc.

Resolve simultaneous signal updates according to event priority

Select this option to control the sequencing of the read event, relative to other simultaneous events in the simulation. If you do not select this option, the

application executes the read event immediately upon detecting the signal-based event that causes it. For details, see “Resolve Simultaneous Signal Updates”. You see this field only if you set **Read from memory upon** to an option other than **Write to memory event**.

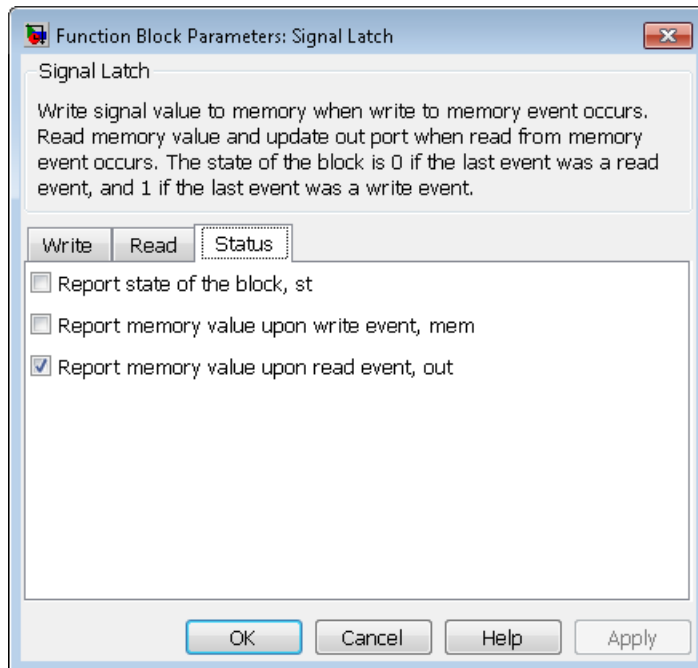
Event priority for reading from memory

The priority of the read event, relative to other simultaneous events in the simulation. For details, see “Specify Event Priorities to Resolve Simultaneous Signal Updates”.

Use of this parameter depends on the following:

- You see this field only if you select **Resolve simultaneous signal updates according to event priority** on this tab.
- If you select **Resolve simultaneous signal updates according to event priority** on both the write tab and the read tab, the software ignores **Event priority for reading from memory**. Instead, the simulation resolves simultaneous signal updates based on only the **Event priority for writing to memory** parameter. In this case, the software executes the write event before the read event.
- If you select the configuration parameter **Prevent duplicate events on multiport blocks and branched signals** in your model, the software uses the **Event priority for writing to memory** parameter to help Simulink to sort blocks in the model. In this case, the software no longer schedules an event that you can view on the SimEvents event calendar.

Status Tab



Report state of the block

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

Report memory value upon write event

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

Report memory value upon read event

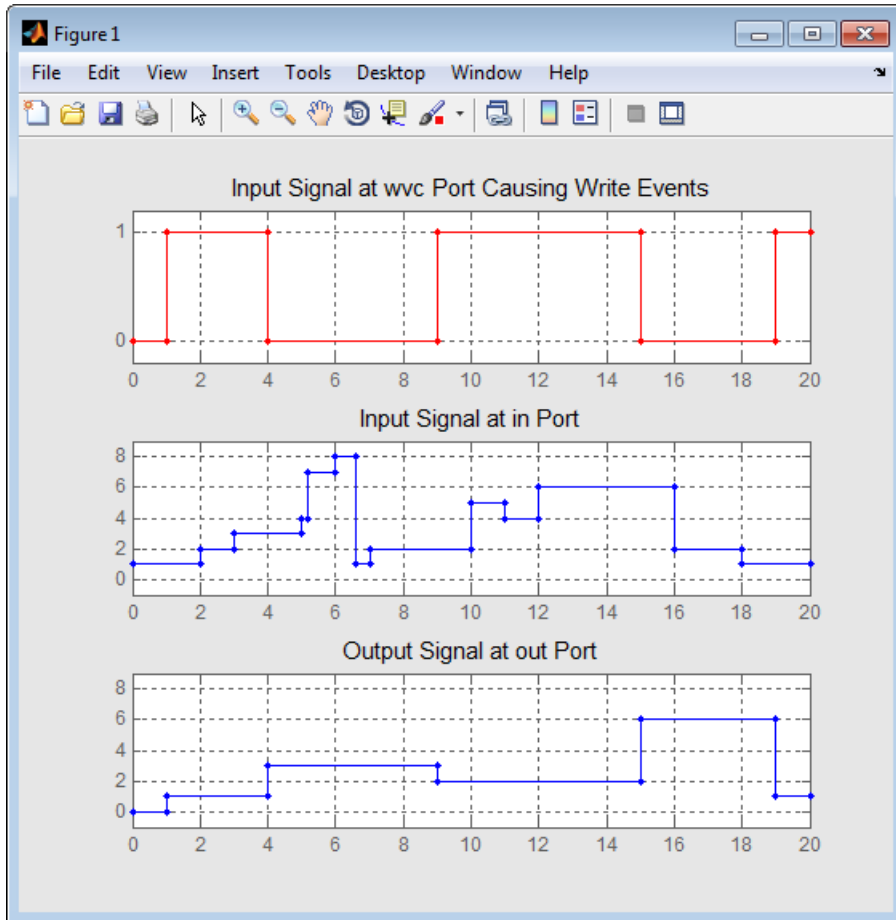
Allows you to use the signal output port labeled **out**.

Examples

Reading from Memory Upon Each Write Event

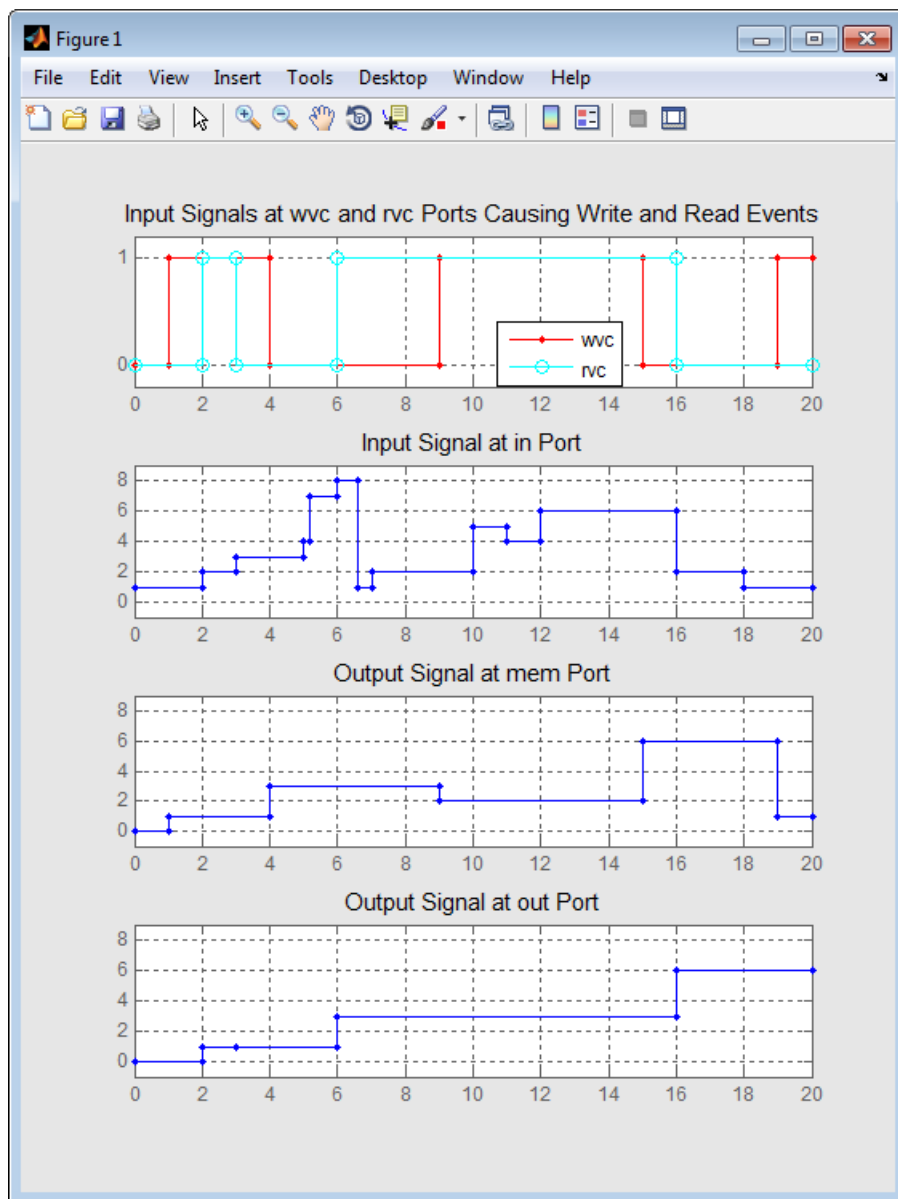
In the plot below, the output signal reflects values of the input signal upon each rising or falling value of the **wvc** signal. Between successive write events, the output signal

maintains the value from the most recent write event. Before the first write event, the output signal is 0 because of the initial memory value.



Independent Read and Write Events

In the plot below, the **mem** signal reflects values of the input signal upon each rising or falling value of the **wvc** signal, while the **out** signal reflects values of the **mem** signal upon each rising or falling value of the **rv** signal.



For examples showing the use of this block in a model, see

- “Create a Random Signal for Switching”
- “Resample a Signal Based on Events”
- “Detect Collisions by Comparing Events”
- “Compound Switching Logic”

See Also

Data Store Memory, Data Store Read, Data Store Write

Signal Scope

Plot data from signal

Library

SimEvents Sinks

Description



This block creates a plot using data from an event-based signal. The data for the vertical axis comes from the signal connected to the block's signal input port labeled **in**.

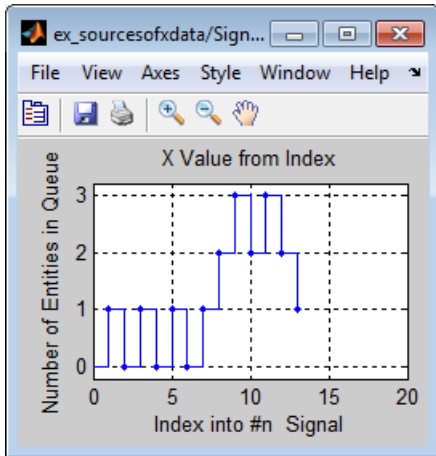
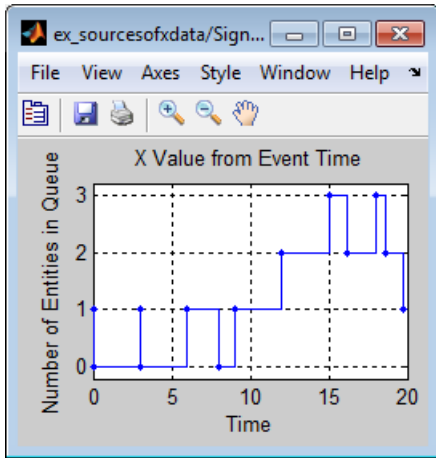
The **Plot type** parameter on the **Plotting** tab determines whether and how the block connects the points that it plots. For details, see “Connections Among Points in Plots”.

Selecting Data for the Horizontal Axis

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

Source of X Data	Description of Plot
Event time	Plot of the in signal versus simulation time. For example, you might use this option to see how the length of a queue changes over time.
Index	Plot of the in signal's successive values against a horizontal axis that represents the index of the values. The signal's first value during the simulation has an index of 1, the signal's second value has an index of 2, and so on. For example, you might use this option for a signal that has zero-duration values, to help determine the exact sequence among values that the signal assumes simultaneously.

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



Ports

Signal Input Ports

Label	Description
in	Signal containing data for the Y axis. This signal must be an event-based signal.

Signal Output Ports

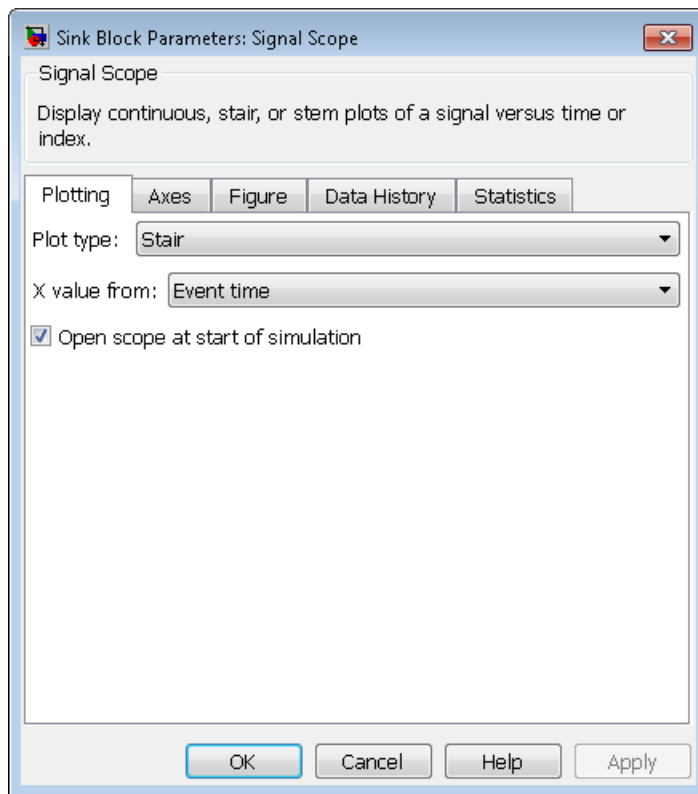
Label	Description
#c	Number of points the block has plotted.

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

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” for details.

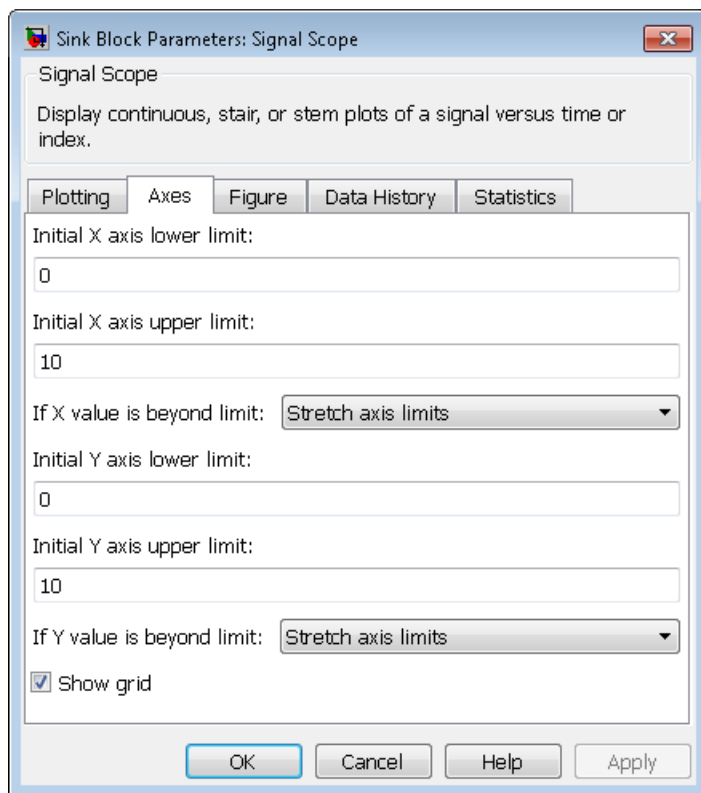
X value from

Source of data for the plot's horizontal axis. See “Selecting Data for the Horizontal Axis” on page 2-233 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 “Vary Axis Limits Automatically”.

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.

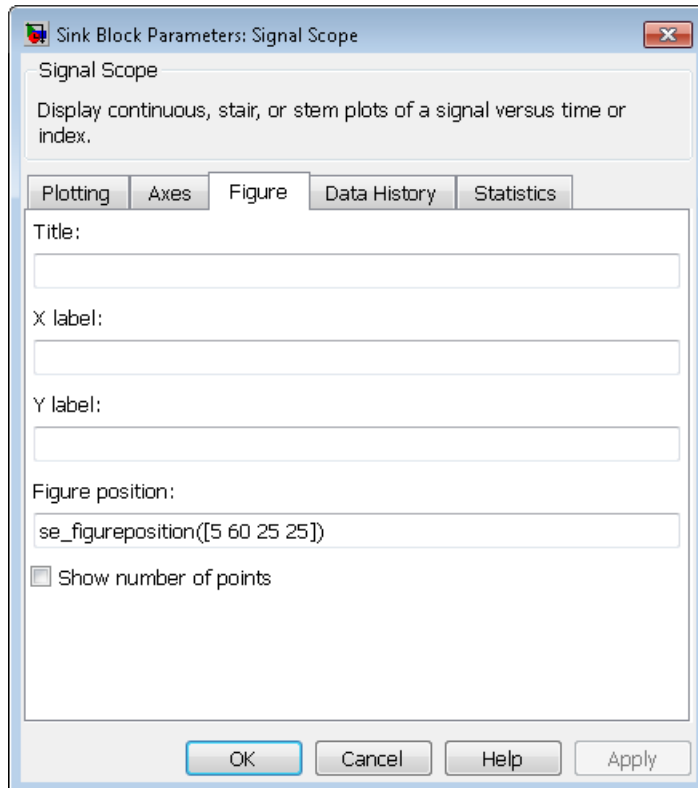
If Y value is beyond limit

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 “Vary Axis Limits Automatically”.

Show grid

Toggles the grid on and off.

Figure Tab



Title

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

Y label

Text that appears to the left of the vertical axis.

X label

Text that appears below the horizontal axis.

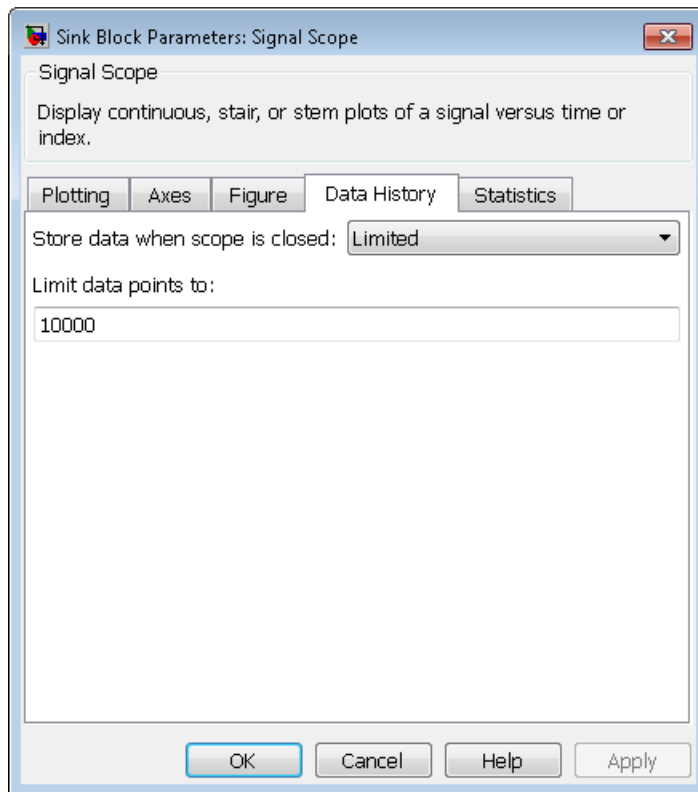
Position

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

Show number of points

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

Data History Tab



Store data when scope is closed

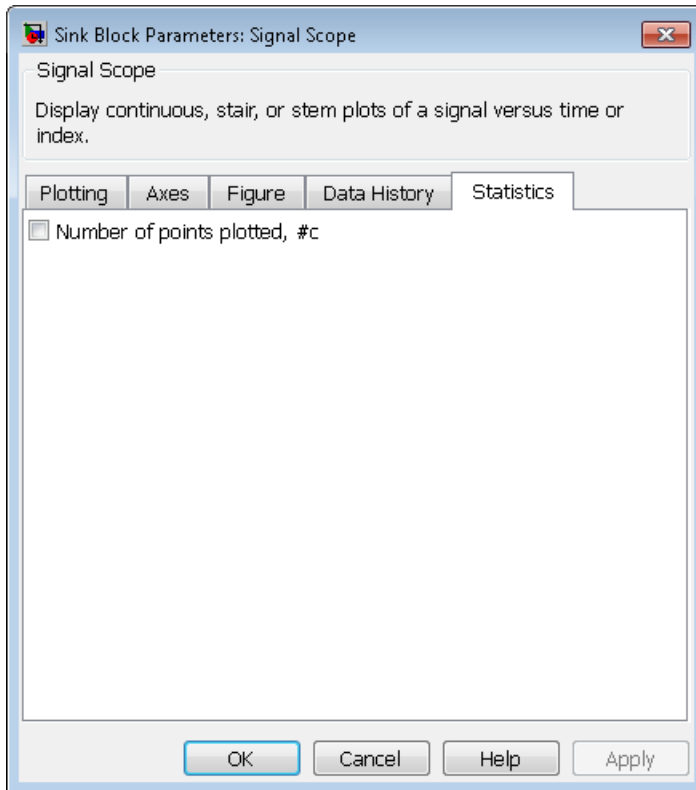
Select **Unlimited** to have the block cache all data for future viewing, **Limited** to cache a portion of the most recent data, and **Disabled** to avoid caching undisplayed data.

Limit data points to

The number of data points the block caches, using the most recent data. You see this field only if you set **Store data when scope is closed** to **Limited**.

Statistics Tab

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



Number of points plotted

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

Examples

- “Build a Discrete-Event Model” and “Observations from Plots”
- “Queue Selection Using a Switch”

See Also

X-Y Signal Scope, Attribute Scope

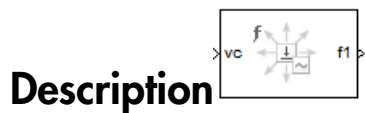
“Choose and Configure Plotting Blocks”

Signal-Based Function-Call Generator

Convert signal-based events into function calls

Library

Generators/Function-Call Generators



This block converts a signal-based event or a function-call input into one or two function calls that you can use to invoke function-call subsystems, Stateflow blocks, or other blocks that accept function-call inputs. You specify the type of event the block translates and whether the block suppresses its output under certain conditions. You can also delay the output function calls by an amount of time that you specify via a parameter or an input signal.

Criteria for Generating Function Calls

The primary criterion, based on the **Generate function call only upon** parameter, is a signal-based event or a function call. By default, the block generates a function call upon each event of the type you specify.

To generate up to two function calls upon each event, select **Generate optional f2 function call**. If the block generates function calls at both the **f1** and **f2** output ports, then it generates the **f1** call first and generates the **f2** call as a subsequent part of the same operation.

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

Ports

Signal Input Ports

Label	Description
t	The delay, in seconds, between the input event and the output function call. A positive value schedules the function call in the future, while a value of zero schedules the function call at the current simulation time. This signal must be an event-based signal. You see this port only if you select Resolve simultaneous signal updates according to event priority , and then set Function-call delay from to Signal port t .
ts	When this signal has an update, the primary criterion is satisfied. This signal must be an event-based signal. You see this port only if you set Generate function call only upon to Sample time hit from port ts .
tr	When this signal has a rising or falling edge, depending on the Trigger type parameter, the primary criterion is satisfied. This signal must be an event-based signal. You see this port only if you set Generate function call only upon to Trigger from port tr .
vc	When this signal increases or decreases, depending on the Type of change in signal value parameter, the primary criterion is satisfied. This signal must be an event-based signal. You see this port only if you set Generate function call only upon to Change in signal from port vc .
fcn	When this signal carries a function call, the primary criterion is satisfied. This signal must be an event-based function call. You see this port only if you set Generate function call only upon to Function call from port fcn . Do not connect this port to an output port from the same instance of this block.
e1	When this signal is 0 or negative, the block does not generate a function call at the f1 output port. This signal must be an event-based signal. You see this input port only if you select Suppress function call f1 if enable signal e1 is not positive .
e2	When this signal is 0 or negative, the block does not generate a function call at the f2 output port. This signal must be an event-based signal. You see this input port only if you select Suppress function call f2 if enable signal e2 is not positive .

Signal Output Ports

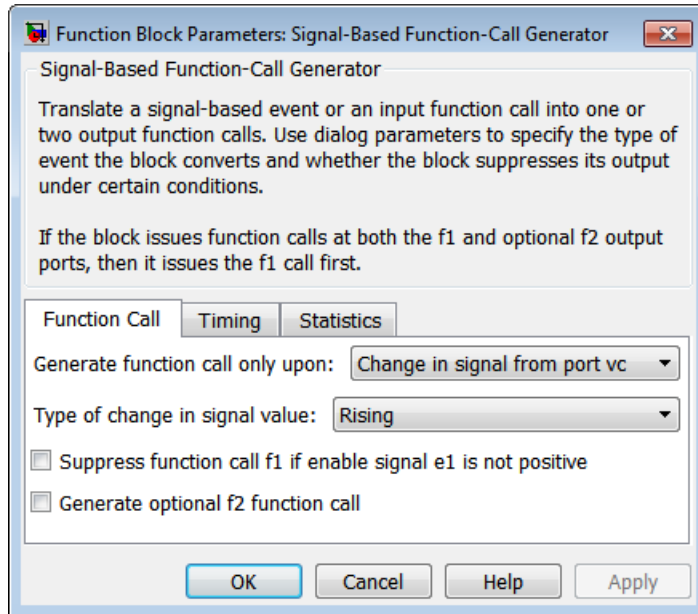
Label	Description	Order of Update
f1	Function call, possibly contingent on e1 input signal	1
f2	Function call, possibly contingent on e2 input signal	2
#f1	Number of function calls the block has generated at the f1 port during the simulation	3
#f2	Number of function calls the block has generated at the f2 port during the simulation	3

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

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

Dialog Box

Function Call Tab



Generate function call only upon

The primary criterion for determining when the block generates a function call. Optional secondary criteria are established by the **Suppress function call...** parameters below.

Trigger type, Type of change in signal value

Trigger type determines whether rising, falling, or either type of trigger edge causes the block to generate a function call. You see this field only if you set **Generate function call only upon** to Trigger from port tr.

Type of change in signal value determines whether rising, falling, or either type of value change causes the block to generate a function call. You see this field only if you set **Generate function call only upon** to Change in signal from port vc.

Suppress function call f1 if enable signal e1 is not positive

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

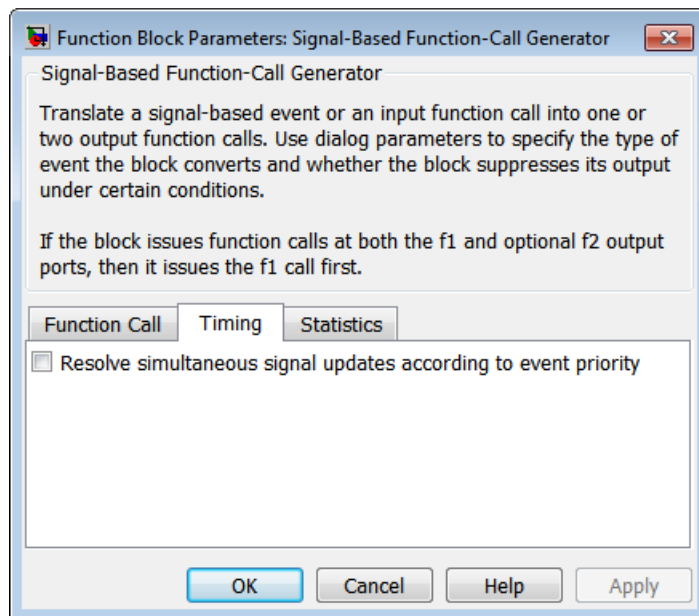
Generate optional f2 function call

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

Suppress function call f2 if enable signal e2 is not positive

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

Timing Tab



Resolve simultaneous signal updates according to event priority

Select this option to control the sequencing of the function-call event, relative to other simultaneous events in the simulation. If you do not select this option, the application issues the function call immediately upon detecting the signal-based event that causes it. For details, see “Resolve Simultaneous Signal Updates”.

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 “Specify Event Priorities to Resolve Simultaneous Signal Updates”.

Use of this parameter depends on the following:

- You see this field only if you select **Resolve simultaneous signal updates according to event priority**.
- If you select the configuration parameter **Prevent duplicate events on multiport blocks and branched signals** in your model, the software uses the **Event priority** parameter to sort blocks in the model. In this case, the software does not schedule an event that you can view on the event calendar.

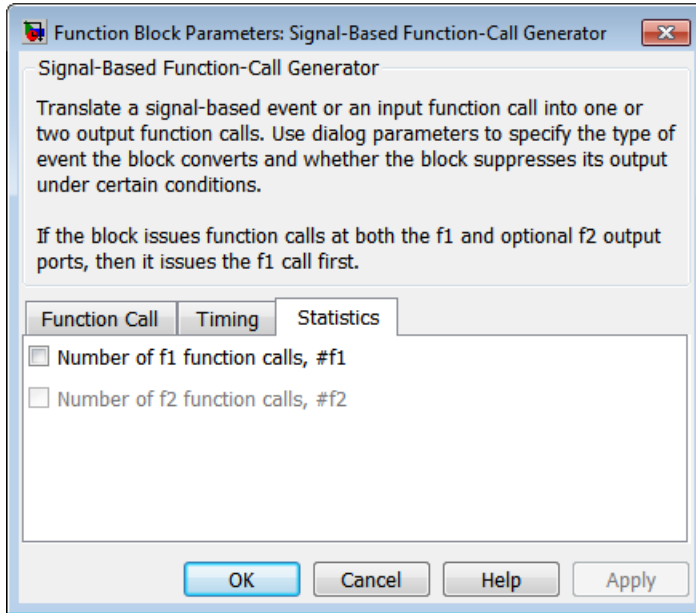
Function-call delay from

Determines whether the delay between the input event and the output function call is computed from a parameter in this dialog box or from an input signal. You see this field only if you select **Resolve simultaneous signal updates according to event priority**.

Function-call time delay

The delay, in seconds, between the input event and the output function call. A positive value schedules the function call in the future, while a value of zero schedules the function call at the current simulation time. You see this field only if you select **Resolve simultaneous signal updates according to event priority**, and then set **Function-call delay from** to Dialog.

Statistics Tab



Number of f1 function calls

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

Number of f2 function calls

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

Examples

- “Detect Changes in the Last-Updated Signal”

See Also

Entity Departure Function-Call Generator

“Manipulate Events”

Signal-Based Function-Call Event Generator

Generate function-call events in response to signal-based events

Library

Generators / Event Generators



Note: The Signal-Based Function-Call Event Generator block will be removed in a future release. Use the Signal-Based Function-Call Generator block instead. To update your model to avoid using obsolete blocks, see [seupdate](#).

This block generates an output function call corresponding to each signal-based event or input function call. You specify the type of event the block responds to. You can use the function call to invoke function-call subsystems, Stateflow blocks, or other blocks that accept function-call inputs.

This block is similar to the Signal-Based Function-Call Generator block, which offers more flexibility.

Ports

Signal Input Ports

Label	Description
ts	When this signal has an update, the block generates a function call. This signal must be an event-based signal. You see this port only if you set Generate function call only upon to Sample time hit from port ts .
tr	When this signal satisfies the specified trigger criteria, the block generates a function call. This signal must be an event-based signal. You see this port only if you set Generate function call only upon to Trigger from port tr .

Label	Description
vc	When this signal satisfies the specified value-change criteria, the block generates a function call. This signal must be an event-based signal. You see this port only if you set Generate function call only upon to Change in signal from port vc .
fcn	When this signal carries a function call, the block generates a function call. This signal must be an event-based function call. You see this port only if you set Generate function call only upon to Function call from port fcn . Do not connect this port to an output port from the same instance of this block.

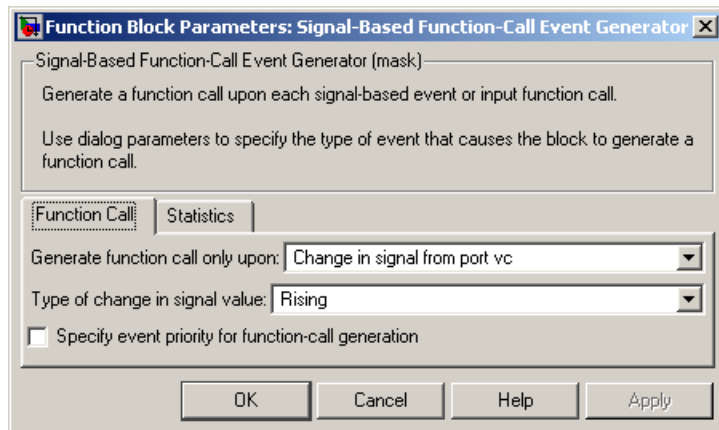
Signal Output Ports

Label	Description	Order of Update
f1	Function-call signal.	1
#f1	Number of function calls the block has generated during the simulation.	2

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

Dialog Box

Function Call Tab



Generate function call only upon

The primary criterion for determining when the block generates a function call. Optional secondary criteria are established by the **Suppress function call...** parameters below.

Trigger type

Determines whether rising, falling, or either type of trigger edge causes the block to generate a function call. You see this field only if you set **Generate function call only upon** to **Trigger** from port **tr**.

Type of change in signal value

Determines whether rising, falling, or either type of value change causes the block to generate a function call. You see this field only if you set **Generate function call only upon** to **Change in signal from port vc**.

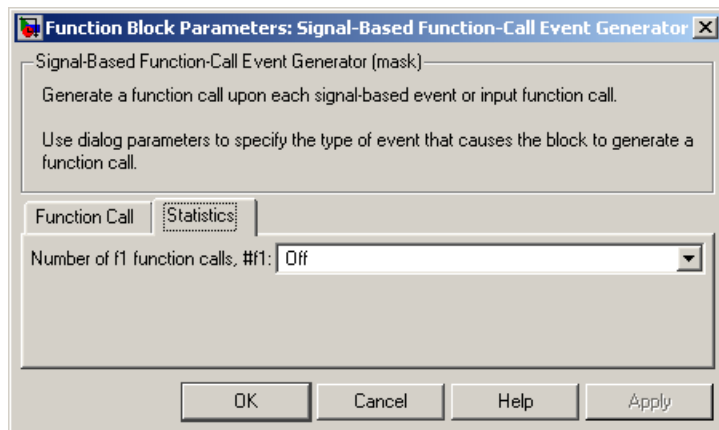
Resolve simultaneous signal updates according to event priority

Select this option to control the sequencing of the function-call event, relative to other simultaneous events in the simulation. If you do not select this option, the application issues the function call immediately upon detecting the signal-based event that causes it. For details, see “Resolve Simultaneous Signal Updates”.

Event priority

The priority of the function-call event, relative to other simultaneous events in the simulation. For details, see “Specify Event Priorities to Resolve Simultaneous Signal Updates”. You see this field only if you select **Resolve simultaneous signal updates according to event priority**.

Statistics Tab



Number of f1 function calls

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

Examples

- “Call a Stateflow Block Upon Changes in Server Contents”
- “Count Events from Multiple Sources”

See Also

Signal-Based Function-Call Generator

“Generate Function-Call Events”

Single Server

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. For an example that uses the **TO** port of a queue block in the same way, see “Use Timeouts to Limit Entity Queuing Time”.

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”.

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 “Preempt an Entity in a Server”.

When the block does not permit preemption, the **IN** port is unavailable whenever this block stores an entity. In this case, the **IN** port becomes available when the entity departs.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities, which will be served.

Signal Input Ports

Label	Description
t	Service time, in seconds, for a newly arrived entity. This signal must be an event-based signal. You see this port only if you set Service time from to Signal port t .

Entity Output Ports

Label	Description
OUT	Port for departing entities that have completed their service time, have not timed out while in this block, and have not been preempted.
P	Port for entities that have been preempted by an arriving entity. This port must not be blocked at the time of preemption.
TO	Port for entities that time out while in this block. You see this port only if you select Enable TO port for timed-out entities . This port must not be blocked when an entity attempts to depart here.

Signal Output Ports

Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#d	Number of entities that have departed from this block via the OUT port since the start of the simulation.	After entity departure via the OUT port	4
#n	Number of entities currently in the block, either 0 or 1.	After entity arrival and after entity departure via the OUT or TO port	3

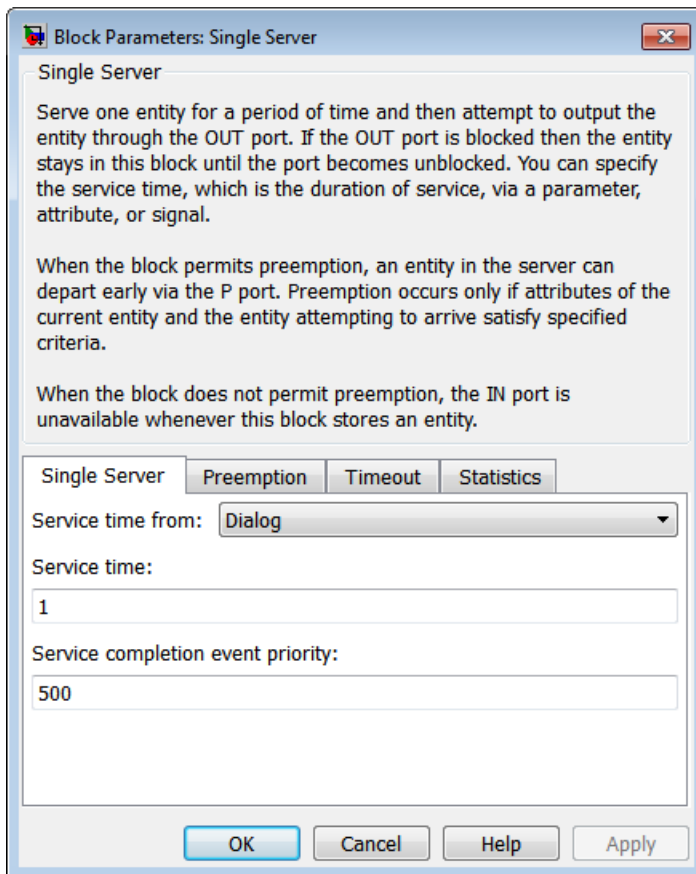
Label	Description	Time of Update When Statistic Is On	Order of Update When Entity Departs
#p	Number of entities that have been preempted from this block since the start of the simulation.	After entity departure via the P port	4
pe	<p>A value of 1 indicates that the block stores at least one entity that has tried and failed to depart. In that case, the entity is a pending entity.</p> <p>A value of 0 indicates that the block does not store any pending entities.</p>	<p>Sample time hit of 1 occurs after the block stores an entity that has tried and failed to depart.</p> <p>Sample time hit of 0 occurs after the departure of the pending entity via any port.</p>	1
w	Sample mean of the waiting times in this block for all entities that have departed from the OUT or TO port. An entity's waiting time might exceed its service time if the OUT port is blocked when the entity completes service.	After entity departure via the OUT or TO port	2
util	Utilization of the server, which is the fraction of simulation time spent storing an entity. At $T=0$, the utilization is 0 or 1 depending on whether the server contains an entity.	Performance considerations cause the block to suppress signal updates until specific occurrences cause updates. In On mode, updates occur after an entity departure via the OUT or TO port, and after an entity arrival. In Upon stop or pause mode, updates occur when the simulation stops or pauses.	2
#to	Number of entities that have timed out from this block since the start of the simulation.	After entity departure via the TO port	4

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

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

Dialog Box

Single Server Tab



Service time from

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

Service time

The service time, in seconds, for all entities. You see this field only if you set **Service time from** to **Dialog**.

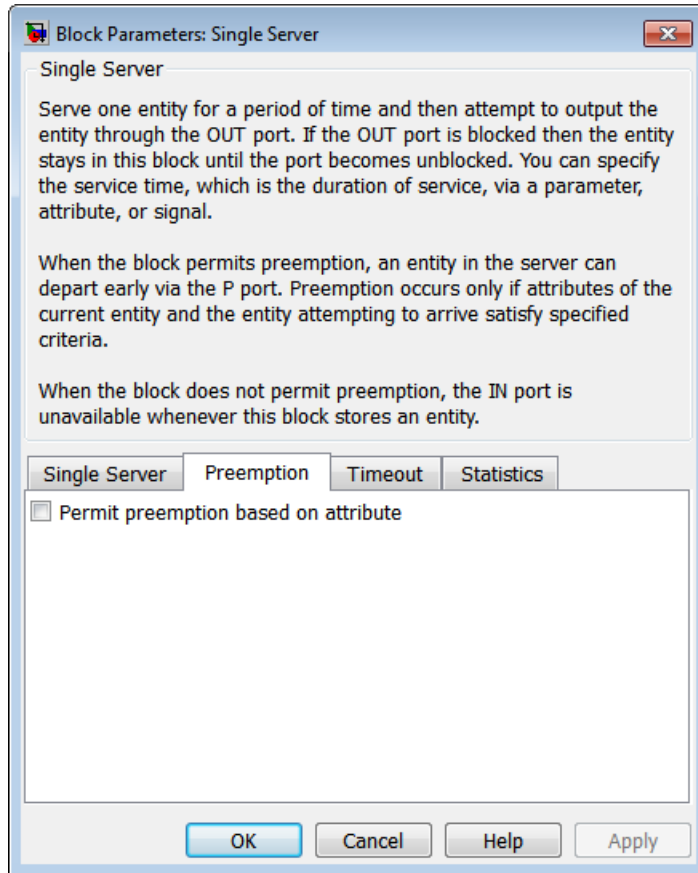
Attribute name

The name of the attribute whose value the block uses as the service time for an entity. You see this field only if you set **Service time from** to **Attribute**.

Service completion event priority

The priority of the service completion event, relative to other simultaneous events in the simulation.

Preemption Tab



Permit preemption based on attribute

If you select this option, the block can replace an entity by a higher priority entity. Otherwise, the block never permits new arrivals when it is storing an entity. Selecting this option also clears the **Average wait, w** check box on the **Statistics** tab and makes that parameter unavailable.

Sorting attribute name

The block uses this attribute to determine whether a new entity can preempt the one in the server. You see this field only if you select **Permit preemption based on attribute**.

Sorting direction

Preemption occurs when the arriving entity has a strictly smaller (**Ascending**) or strictly larger (**Descending**) value of the attribute named above, compared to the attribute value of the entity in the server. You see this field only if you select **Permit preemption based on attribute**.

Write residual service time to attribute

If you select this option, a preemption event causes the block to set an attribute in the preempted entity. The attribute value is the remaining service time the entity would have required if it had not been preempted. You see this field only if you select **Permit preemption based on attribute**.

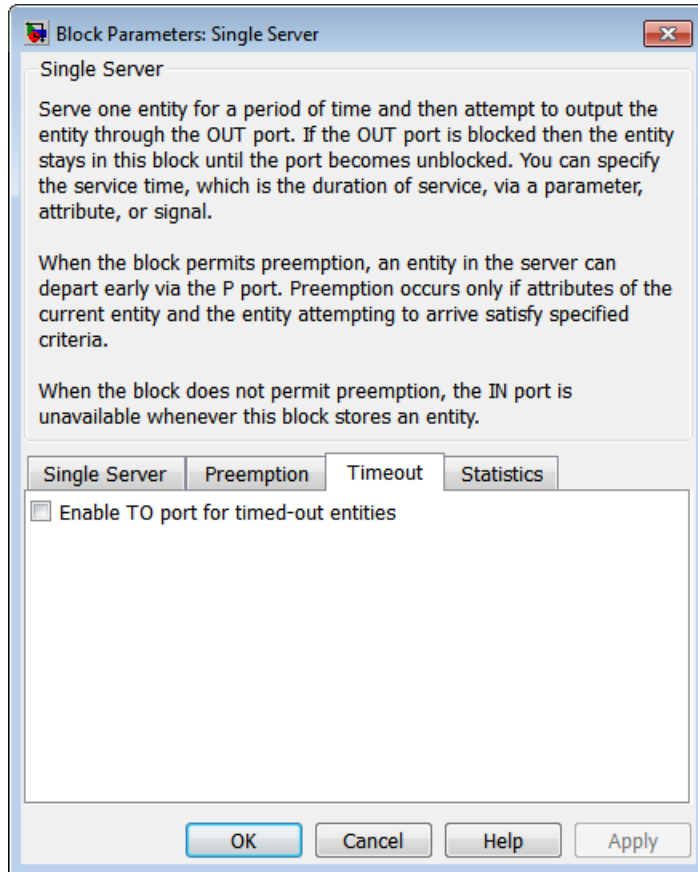
Residual service time attribute name

The name of the attribute the block uses when recording the residual service time of entities. You see this field only if you select **Write residual service time to attribute**.

Create attribute if not present

Selecting this option enables the block to define a new attribute for the residual service time. Otherwise, the block issues an error if the attribute named above does not already exist. You see this field only if you select **Write residual service time to attribute**.

Timeout Tab



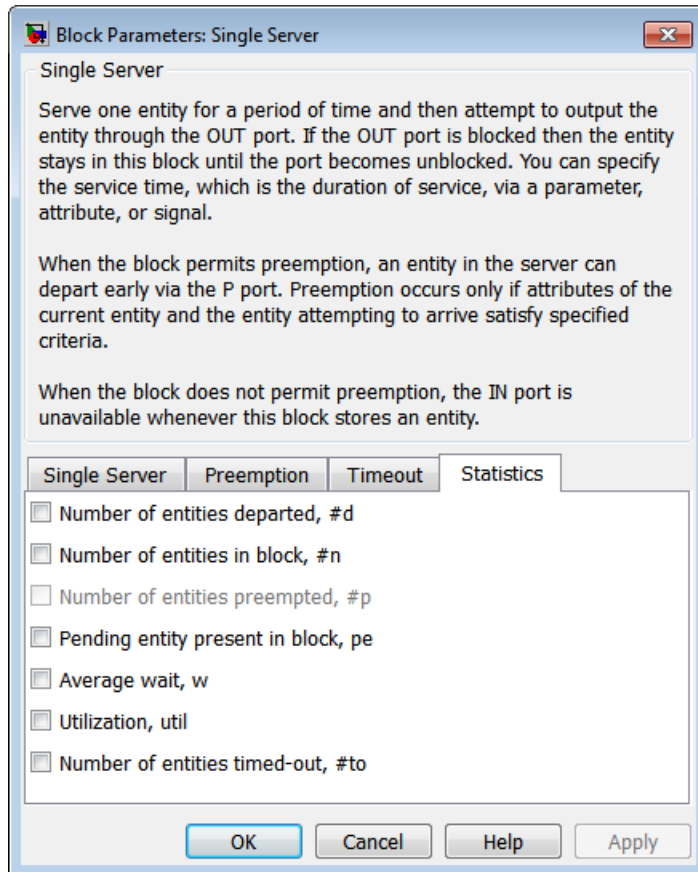
Enable TO port for timed-out entities

This option becomes relevant if an entity times out while in this block. Selecting this option provides a **TO** entity output port through which the timed-out entity departs. If you clear this option in a model that uses timeouts, see the **If entity has no destination when timeout occurs** parameter of the Schedule Timeout block.

Statistics Tab

These parameters determine whether certain ports produce data throughout the simulation, produce data only when you stop or pause the simulation, or are omitted

from the block. For descriptions of the affected ports, see the preceding table, “Signal Output Ports”.



Number of entities departed

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

Number of entities in block

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

Number of entities preempted

Allows you to use the signal output port labeled **#p**. This field is available only if you select the **Permit preemption based on attribute** option on the **Preemption** tab.

Pending entity present in block

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

Average wait

Allows you to use the signal output port labeled **w**. This field is available only if you clear the **Permit preemption based on attribute** option on the **Preemption** tab.

Utilization

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

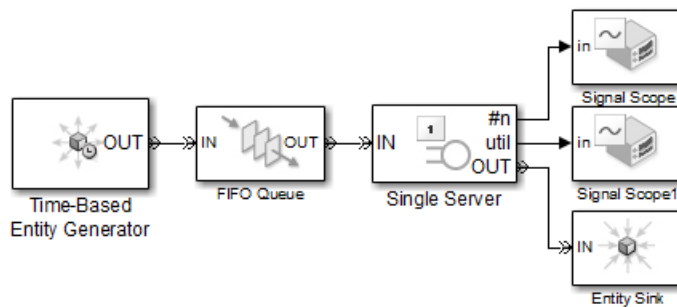
Number of entities timed out

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

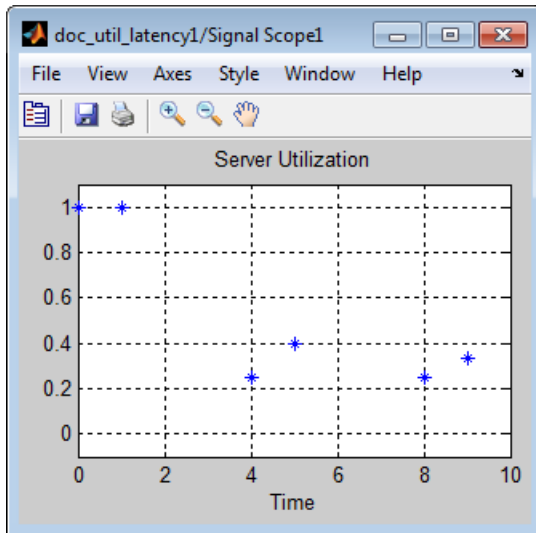
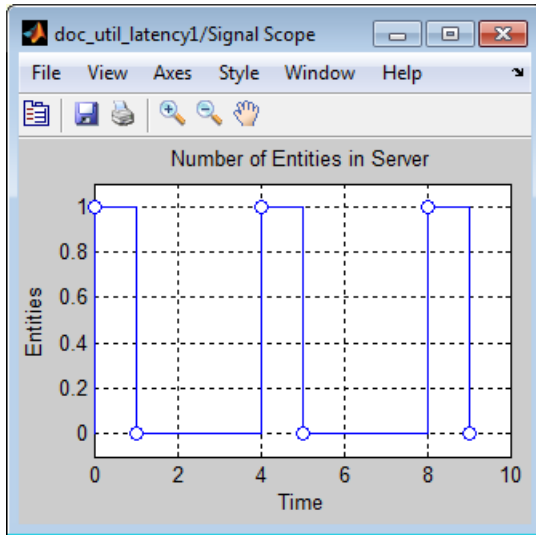
Examples

- “Build a Discrete-Event Model”
- “Select the First Available Server”
- “Constructs Involving Queues and Servers”
- “Use a Signal or an Attribute”
- “Preemption by High-Priority Entities”
- “Control Joint Availability of Two Servers”
- “Synchronize Service Start Times with the Clock”

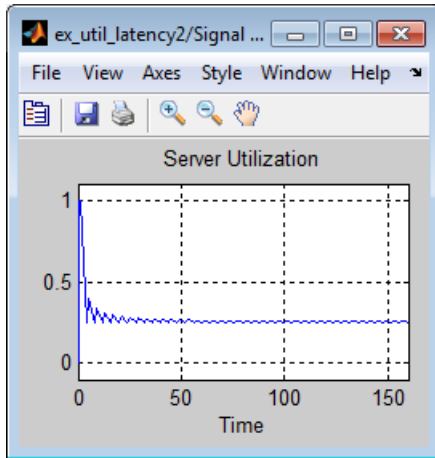
The following example illustrates the timing of updates of the **util** signal, as described in Signal Output Ports.



The server has idle periods that reduce its utilization. However, the server block recomputes the **util** signal only when the number of entities in the server changes. While the definition of utilization says that the utilization is less than 1 at time 3, the **util** signal remains at its previous value of 1 until the next entity arrives at time 4.



In a longer simulation, the differences in the value of **util** compared to its theoretical definition become less pronounced.



See Also

N-Server, Infinite Server

“Servers in SimEvents Models”

Start Timer

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 “Measure Point-to-Point Delays”.

Ports

Entity Input Ports

Label	Description
IN	Port for arriving entities.

Entity Output Ports

Label	Description
OUT	Port for departing entities, which have named timers attached to them.

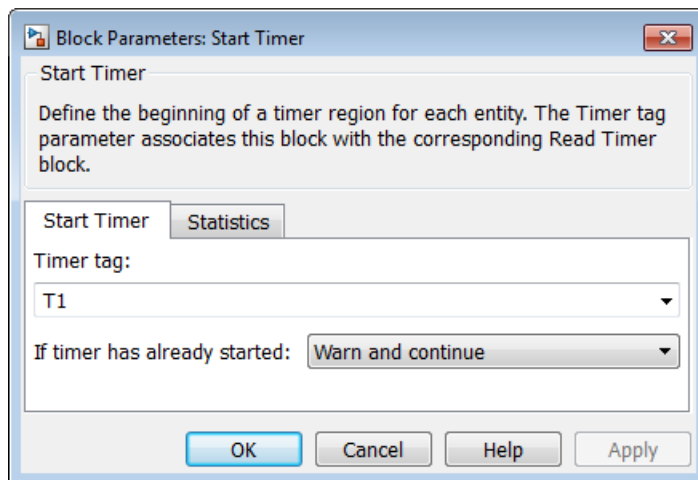
Signal Output Ports

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

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

Dialog Box

Start Timer Tab



Timer tag

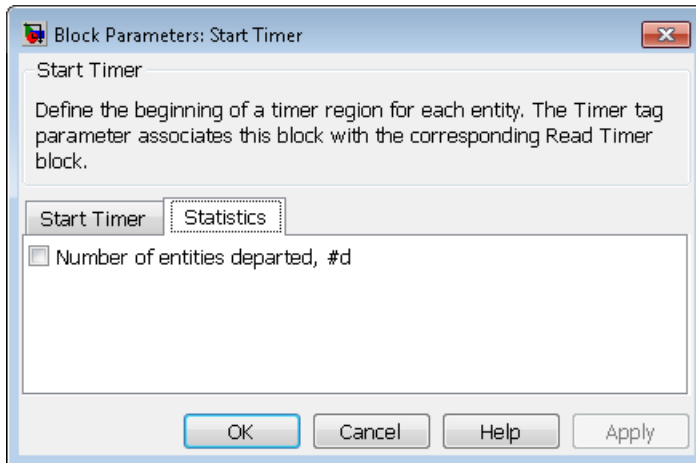
Name of the timer to associate with each entity. Enter a new timer tag, or restart a previous timer by choosing it in the drop-down list.

If timer has already started

Behavior of the block if an arriving entity already has a timer with the specified timer tag.

Statistics Tab

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



Number of entities departed

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

Examples

- “Basic Procedure for Using Timer Blocks”
- “Time Multiple Entity Paths with One Timer”
- “Restart a Timer from Zero”
- “Time Multiple Processes Independently”

See Also

Read Timer

“Measure Point-to-Point Delays”

Time-Based Entity Generator

Generate entities using intergeneration times from signal or statistical distribution

Library

Generators / Entity Generators



This block is designed to generate entities using intergeneration times that satisfy criteria that you specify. The intergeneration time is the time interval between two successive generation events.

Intergeneration Times	Value of Generate entities upon Parameter
Distributed according to various parameters in the block dialog box	Intergeneration time from dialog
Specified using an input signal that the block reads at the start of the simulation and each time it generates an entity	Intergeneration time from port t

For details about these options, see “Specify Intergeneration Times for Entities”.

Responding to Blockage at the Entity Output Port

You can choose how this block responds when it generates an entity that the subsequent entity input port is not available to accept:

- If you set **Response when blocked** to **Error**, the simulation halts with an error message.
- If you set **Response when blocked** to **Pause generation**, this block holds the entity, which becomes a pending entity. The block does not schedule another entity

generation event yet. The **Response when unblocked** parameter determines what the block does next:

- If you set **Response when unblocked** to **Immediate restart**, after this block learns that the subsequent port is available, the pending entity departs. After the pending entity departs, this block schedules the generation of the next entity.
- If you set **Response when unblocked** to **Delayed restart**, upon learning that the subsequent port is available, this block schedules an event of type **DelayedRestart**. The event time is the current time plus the same intergeneration time the block used when generating the pending entity. When the block executes the event, the pending entity attempts to depart.

Use the **Delayed restart** option if you want to:

- Keep the arrival process memoryless, when **Distribution** is **Exponential**.
- Prevent correlation among multiple instances of this block if they become unblocked simultaneously.

For an example, see “Example: Responding to Blockage” on page 2-277.

Ports

Signal Input Ports

Label	Description
t	Time interval between generation events of the current entity and the next entity. The block reads the value after the current entity departs and the block updates its output signals, if any. If you do not select Generate entity at simulation start , then the block also reads the value of this signal at the start of the simulation. This signal must be an event-based signal. You see this port only if you set Generate entities upon to Intergeneration time from port t.

Entity Output Ports

Label	Description
OUT	Port through which generated entities depart.

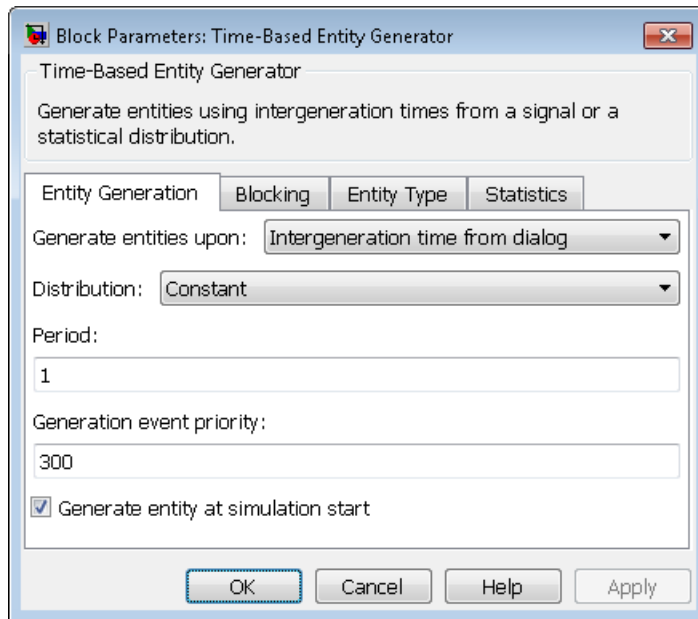
Signal Output Ports

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

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

Dialog Box

Entity Generation Tab



Generate entities upon

Determines where the block gets instructions about when to generate entities.

Distribution

The statistical distribution of intergeneration times. You see this field only if you set **Generate entities upon** to Intergeneration time from dialog.

Period

The time interval between entity generations, in seconds. You see this field only if you set **Generate entities upon** to Intergeneration time from dialog and set **Distribution** to Constant.

Initial seed

A nonnegative integer that initializes the random number generator. You see this field only if you set **Generate entities upon** to Intergeneration time from dialog and set **Distribution** to Uniform or Exponential.

Minimum

The lower endpoint, in seconds, of the interval over which the distribution is uniform. This field appears only if you set **Generate entities upon** to Intergeneration time from dialog and set **Distribution** to Uniform.

Maximum

The upper endpoint, in seconds, of the interval over which the distribution is uniform. This field appears only if you set **Generate entities upon** to Intergeneration time from dialog and set **Distribution** to Uniform.

Mean

The expected value of the exponential distribution. You see this field only if you set **Generate entities upon** to Intergeneration time from dialog and set **Distribution** to Exponential.

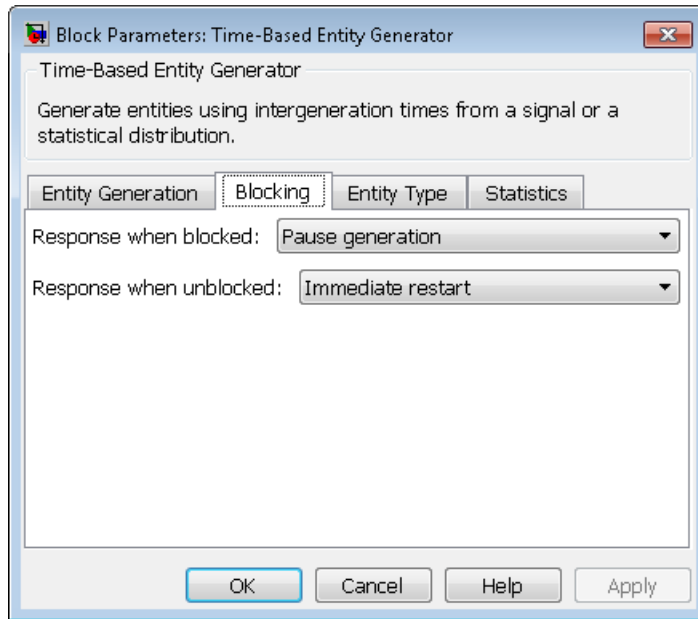
Generation event priority

The priority of the entity-generation event, relative to other simultaneous events in the simulation.

Generate entity at simulation start

If you select this option, the block generates the first entity when the simulation begins and the second entity at the first intergeneration time. Otherwise, the block generates the first entity at the first intergeneration time.

Blocking Tab



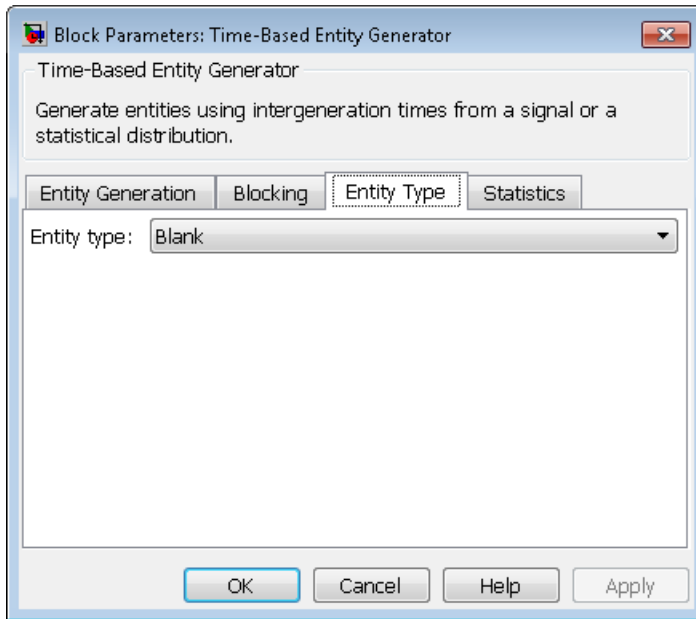
Response when blocked

Determines how the block responds if a generated entity cannot depart immediately because the entity input port of the subsequent block is unavailable; see “Responding to Blockage at the Entity Output Port” on page 2-269.

Response when unblocked

Determines entity generation behavior if the entity input port of the subsequent block is available again after a prior blockage; see “Responding to Blockage at the Entity Output Port” on page 2-269.

Entity Type Tab

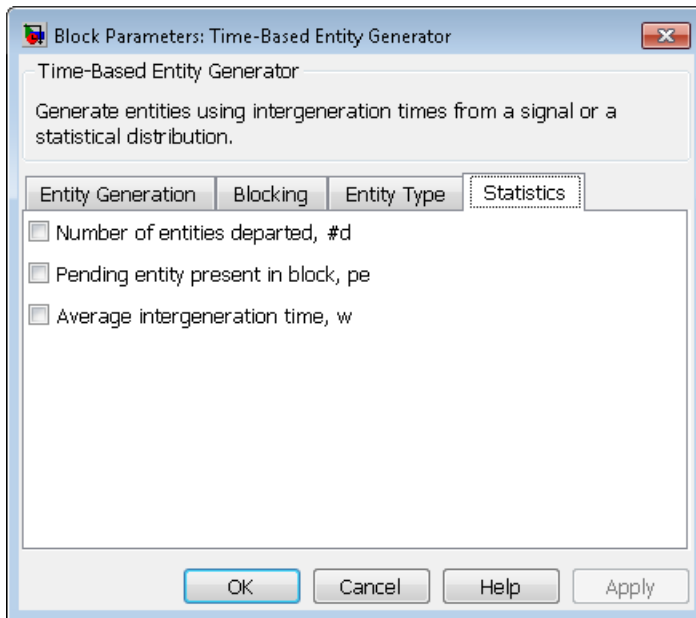


Entity type

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

Statistics Tab

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



Number of entities departed

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

Pending entity present in block

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

Average intergeneration time

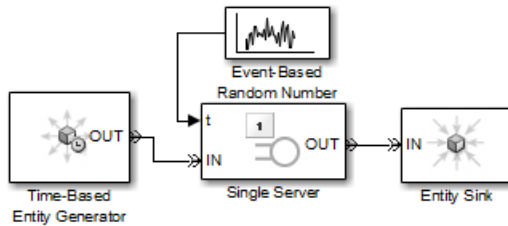
Allows you to use the signal output port labeled **w**.

Examples

- “Build a Discrete-Event Model”
- “Using Random Intergeneration Times in a Queuing System”
- “Use a Step Function as Intergeneration Time”
- “Use an Arbitrary Discrete Distribution as Intergeneration Time”
- “Specify Generation Times for Entities”

Example: Responding to Blockage

To illustrate the blockage options, consider a Time-Based Entity Generator block followed by a Single Server block, then followed by an Entity Sink block.



Suppose the block configurations have these characteristics:

- The entity generator has **Response when blocked** set to **Pause** generation.
- The entity generator generates the first entity at $T=1$ and uses an intergeneration time of 1 s.
- The service times for the first three entities in the server are 1.5, 2.2, and 1.8.

The following tables indicate how the **Response when unblocked** values affect the behavior in the simulation.

Immediate Restart

Time (s)	Behavior
1	Entity generator generates and outputs the first entity to the server. The entity input port of the server becomes unavailable. The first entity is in service until $T=1+1.5=2.5$.
2	Entity generator generates the second entity and holds it because the OUT port is blocked.
2.5	First entity departs from the server. The entity input port of the server becomes available and the second entity advances from the entity generator to the server. The entity input port of the server then becomes unavailable. The second entity is in service until $T=2.5+2.2=4.7$. The entity generator schedules the next generation for $T=2.5+1=3.5$.
3.5	Entity generator generates the third entity, and holds it because the OUT port is blocked.

Time (s)	Behavior
4.7	Second entity departs from the server. The entity input port of the server becomes available and the third entity advances from the entity generator to the server. The entity input port of the server then becomes unavailable. The third entity is in service until $T=4.7+1.8=6.5$. The entity generator schedules the next generation for $T=4.7+1=5.7$.

Delayed Restart

Time (s)	Behavior
1	Entity generator generates the first entity. The entity advances to the server. The entity input port of the server becomes unavailable. The entity is in service until $T=1+1.5=2.5$.
2	Entity generator generates the second entity. The entity becomes a pending entity because the OUT port is blocked.
2.5	First entity departs from the server. The entity input port of the server becomes available. The entity generator schedules a delayed restart event for the second entity at $T=2.5+1=3.5$.
3.5	The second entity advances from the entity generator to the server. The entity input port of the server then becomes unavailable. The second entity is in service until $T=3.5+2.2=5.7$. The entity generator schedules the next generation for $T=3.5+1=4.5$.
4.5	Entity generator generates the third entity. The entity becomes a pending entity because the OUT port is blocked.
5.7	Second entity departs from the server. The entity input port of the server becomes available. The entity generator schedules a delayed restart event for the third entity at $T=5.7+1=6.7$.
6.7	The third entity advances from the entity generator to the server. The entity input port of the server then becomes unavailable. The third entity is in service until $T=6.7+1.8=8.5$. The entity generator schedules the next generation for $T=6.7+1=7.7$.

See Also

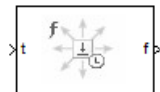
Event-Based Entity Generator, Entity Sink

Time-Based Function-Call Generator

Generate function-call events in a time-based manner.

Library

Generators/Function-Call Generators



Description

This block generates function-call events, either once at the start of simulation, using an intergeneration period that you specify in the block dialog box, or using a signal connected to the input port. The intergeneration period is the time interval between two successive generation events.

You can set the **Event generation mode** parameter of the block to one of three values. The block determines the intergeneration period differently for each value of the **Event generation mode** parameter that you choose.

Value of Event generation mode Parameter	Intergeneration Period
Only at simulation start	None, because only one event is generated
Period from dialog	Specified in the Period parameter of the block dialog box
Period from port	Specified using an input signal t that the block reads at the start of the simulation and each time it generates an entity

If you select the configuration parameter **Prevent duplicate events on multiport blocks and branched signals** in your model, the Time-Based Function-Call Generator block is compatible with all other blocks from SimEvents version 4.0 (R2011b), or later.

For more information about:

- Identifying blocks in your model from releases prior to R2011b, see “Visual Appearance of Legacy SimEvents Blocks” in the SimEvents documentation.
- The configuration parameter **Prevent duplicate events on multiport blocks and branched signals**, see “Prevent duplicate events on multiport blocks and branched signals” in the SimEvents documentation.

Ports

Signal Input Ports

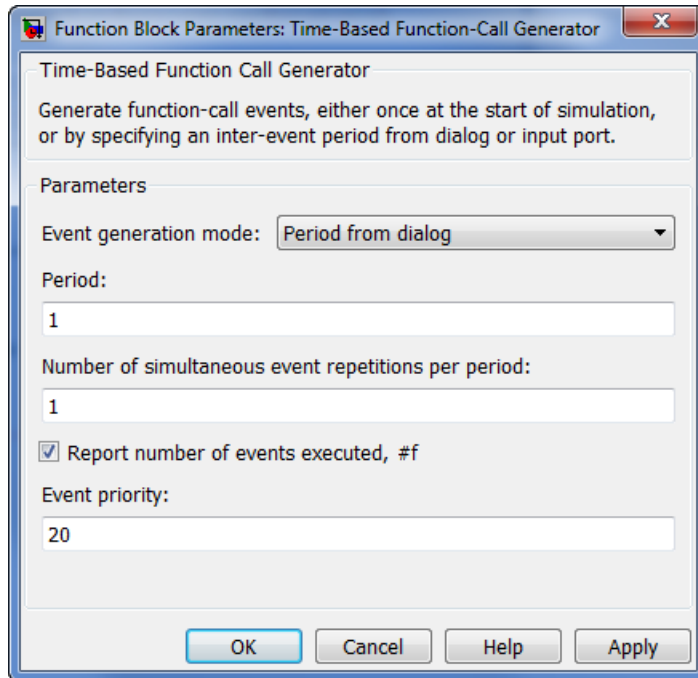
Label	Description
t	Time interval between generation of the current function-call event and the next function-call event. You see this port only if you set the Event generation mode parameter to Period from port in the block dialog box. At the start of the simulation and each time the block updates its output signals, the block reads the value at the input port, or if the preceding block is an Event-Based Random Number or Event-Based Sequence block, actively requests an updated input value. The signal connected to the input port must be an event-based signal.

Signal Output Ports

Label	Description
f	Port through which generated function-call signals depart.
#f	Number of function-call events that have been executed by the block. You only see this port if you select the Report number of events executed, #f check box in the block dialog box.

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

Dialog Box



Event generation mode

Determines the mode that the block uses to generate function-call events.

Period

The time interval between generation of successive function-call events, in seconds. You see this field only if you set the **Event generation mode** parameter to **Period from dialog** in the block dialog box.

Number of simultaneous event repetitions per period

The number of simultaneous function-call events that the block generates in each period. Use this parameter to generate a function-call with multiple iterations.

Report number of events executed, #f

Determines if the number of function-call events that have executed by the block is made available via a signal output port on the block.

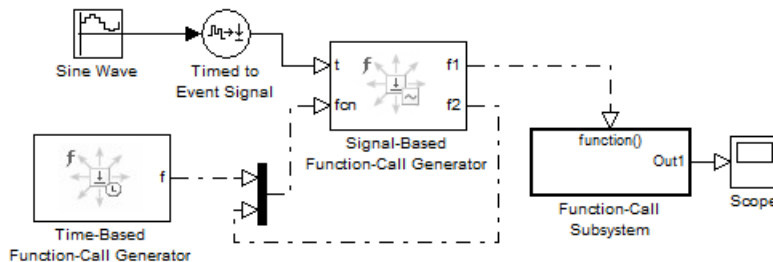
Event priority

The priority of the function-call event relative to other simultaneous events in the simulation.

Examples

Seed Event Generation

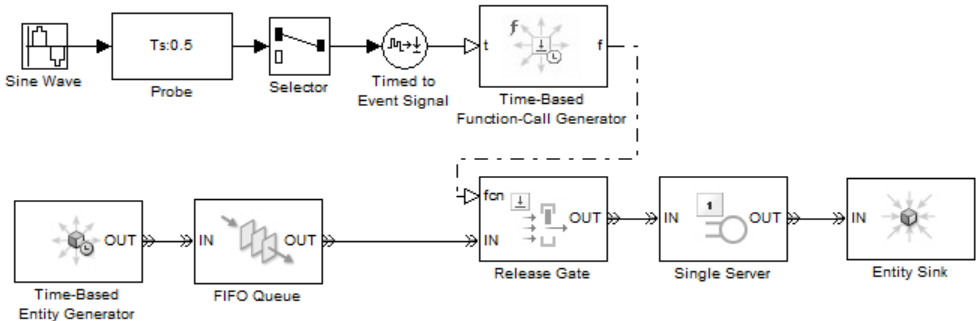
To generate a seed event in your model at simulation time $T=0$, you can use the Time-Based Function-Call Generator block. A seed event is an initial impulse that models with certain block configurations require to update the outputs of their blocks and to start generating events.



In this example, the **Event generation mode** parameter of the Time-Based Function-Call Generator block is set to **Only at simulation start**. At simulation time $T=0$, the Time-Based Function-Call Generator block produces an initial function-call event — or seed event — to the Signal-Based Function-Call Generator block. This seed event causes the Signal-Based Function-Call Generator block to update its outputs. After simulation time $T=0$, the simulation continues to update the outputs of the Signal-Based Function-Call Generator block. The model is now self-sustaining.

Time Synchronization

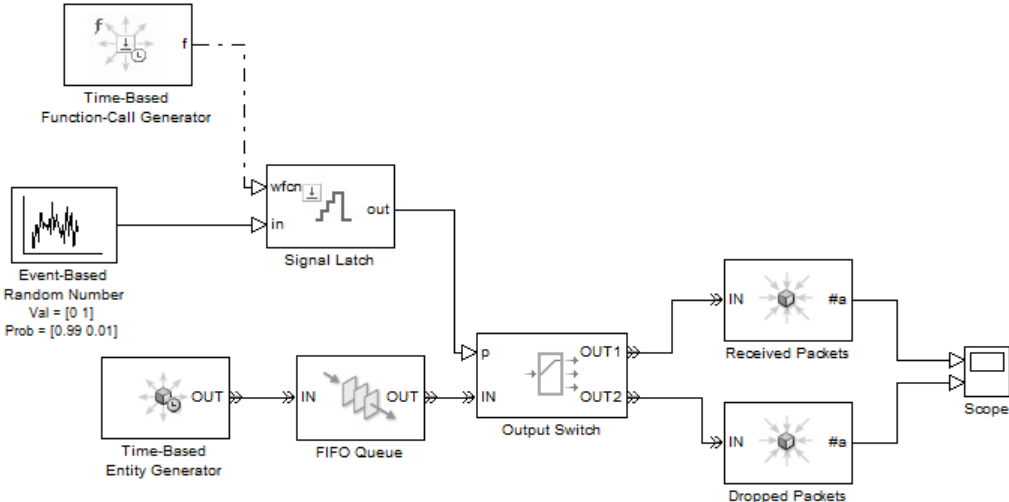
When you want to synchronize events with a time-based process outside the SimEvents domain, use the Time-Based Function-Call Generator block.



In this model, the **Event generation mode** parameter of the Time-Based Function-Call Generator block is set to **Period** from port. The Probe block detects the sample time of a sine wave signal and connects it to the input port **t** of the Time-Based Function-Call Generator block. The value at the input port **t** determines the period — or time delay — between successive function-call events generated by the block and is used to control entity advancement through the Release Gate block.

Discrete-Event Model with Statistical Distribution

You can use the Time-Based Function-Call Generator block to model a discrete-event system that is driven by a time-based process drawn from a statistical distribution.



In this model, using a statistical distribution, the Event-Based Random Number block generates random values. The Time-Based Generator block periodically produces a function-call event to store the current output value of the Event-Based Random Number block in the Signal Latch block. This value is used, in turn, to select the output port of the Output Switch block. Based on the statistical distribution specified in this model, the value that is output by the Signal Latch block has a 99% probability of being **1**. When this situation is the case, the *first* output port of the Output Switch block is selected, and an entity advances as a received packet. Conversely, 1% of entities advance as dropped packets.

See Also

Signal-Based Function-Call Generator

“Function Calls”

“Generate Function-Call Events”

“Manipulate Events”

Timed to Event Function-Call

Convert time-based function call to event-based function call

Library

Gateways

Description

This block converts a scalar time-based function call into an event-based function call. The output signal is almost identical to the input signal, except that the output can be an input to a block that requires an event-based function-call input signal.

Ports

Signal Input Ports

Label	Description
None	Time-based function-call signal.

Signal Output Ports

Label	Description
None	Event-based function-call signal.

See Also

Event to Timed Function-Call

“Time-Based and Event-Based Signal Conversion”

Timed to Event Signal

Convert time-based signal to event-based signal

Library

Gateways

Description

This block converts a time-based data signal into an event-based data signal. The value of the output signal is identical to that of the input signal. The output signal can be an input to a block that requires an event-based input signal.

Ports

Signal Input Ports

Label	Description
None	Time-based signal. The signal can have any fixed dimension, complexity, or data type.

Signal Output Ports

Label	Description
None	Event-based signal

The initial output value is the same as the initial input value.

See Also

Event to Timed Signal

“Time-Based and Event-Based Signal Conversion”

X-Y Attribute Scope

Plot data from two attributes of arriving entities

Library

SimEvents Sinks



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”.

Ports

Entity Input Ports

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

Entity Output Ports

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

Signal Output Ports

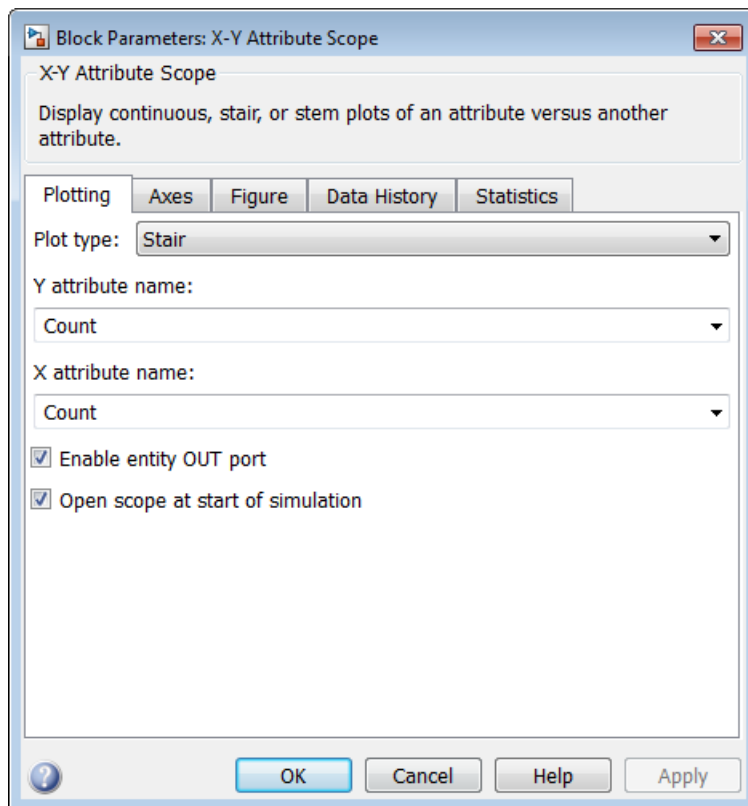
Label	Description
#a	Number of entities that have arrived at the block since the start of the simulation.

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

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” 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.

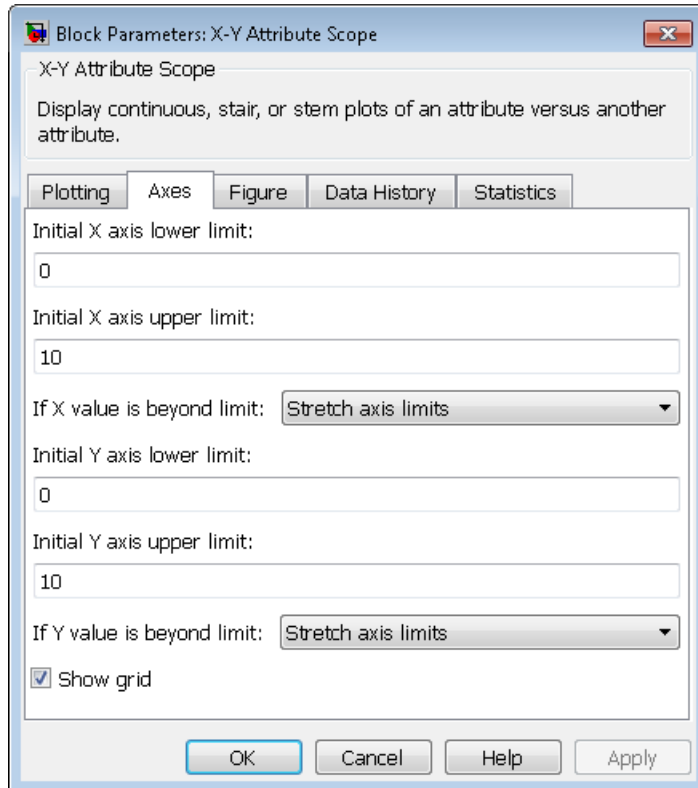
Enable entity OUT port

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 “Vary Axis Limits Automatically”.

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.

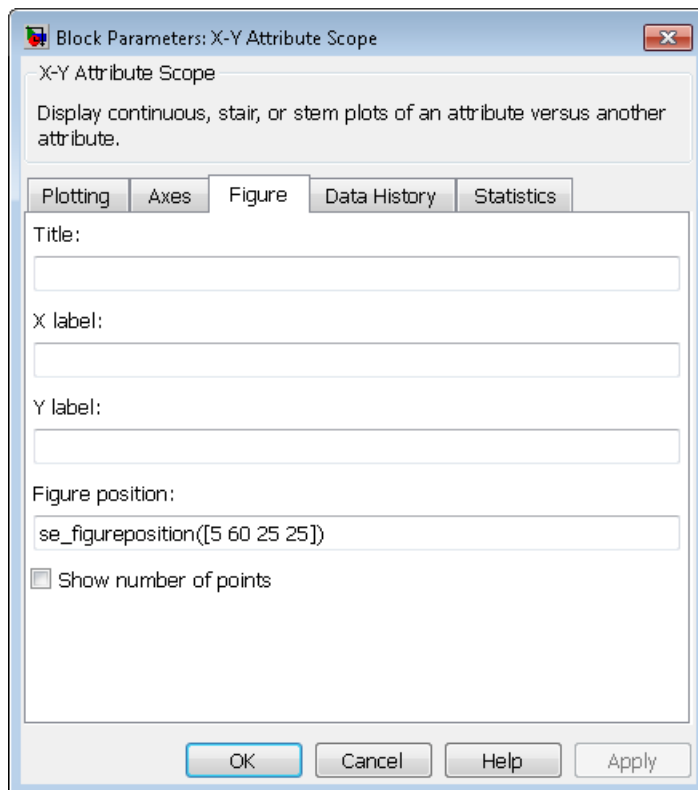
If Y value is beyond limit

Determines how the plot changes if one or more Y values are not within the limits shown on the Y axis. For details, see “Vary Axis Limits Automatically”.

Show grid

Toggles the grid on and off.

Figure Tab



Title

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

Y label

Text that appears to the left of the vertical axis.

X label

Text that appears below the horizontal axis.

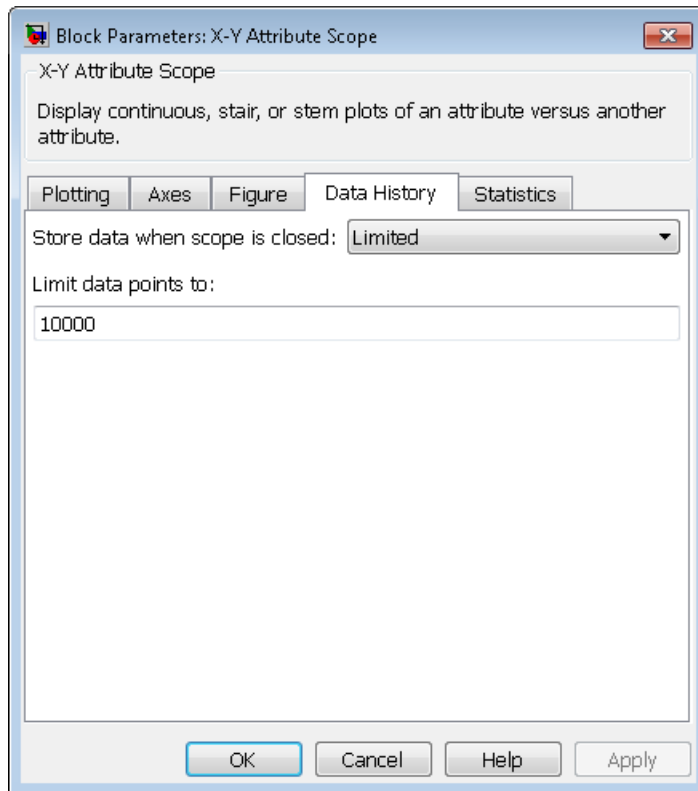
Figure Position

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

Show number of entities

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

Data History Tab



Store data when scope is closed

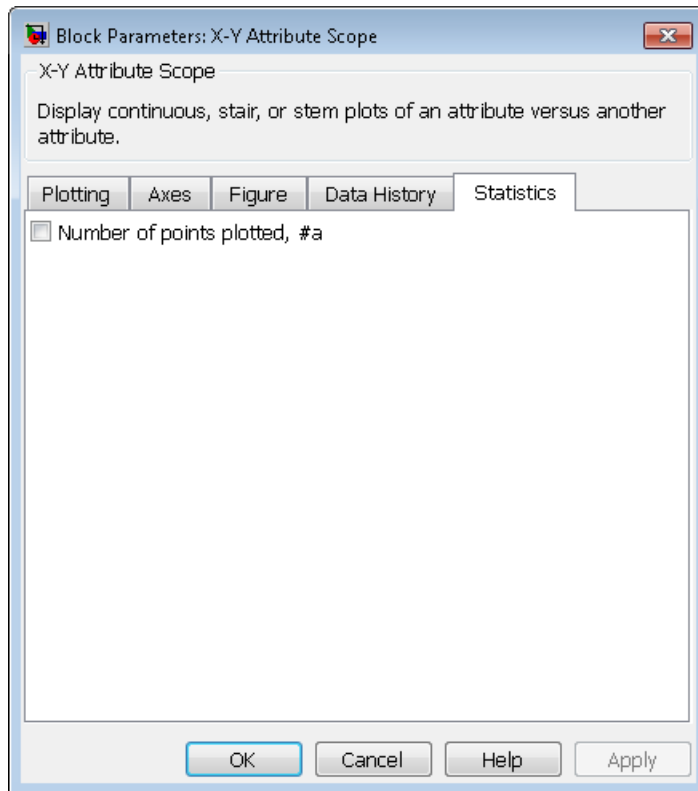
Select **Unlimited** to have the block cache all data for future viewing, **Limited** to cache a portion of the most recent data, and **Disabled** to avoid caching undisplayed data.

Limit data points to

The number of data points the block caches, using the most recent data. You see this field only if you set **Store data when scope is closed** to **Limited**.

Statistics Tab

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

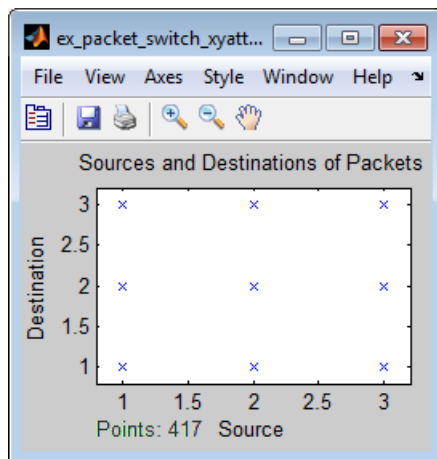


Number of points plotted

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

Examples

You can modify the example in “Model a Packet Switch” 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

“Choose and Configure Plotting Blocks”, “Access Entity Attributes”

X-Y Signal Scope

Plot data from two signals

Library

SimEvents Sinks



Description

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

Ports

Signal Input Ports

Label	Description
in	Signal containing data for Y axis. This signal must be an event-based signal.
x	Signal containing data for X axis. This signal must be an event-based signal.

Signal Output Ports

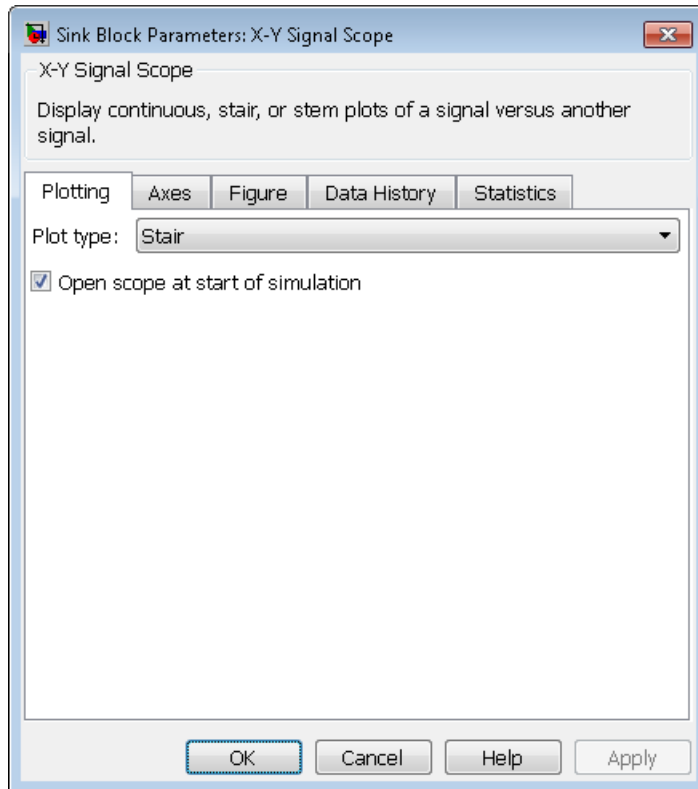
Label	Description
#c	Number of points the block has plotted.

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

Dialog Box

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

Plotting Tab



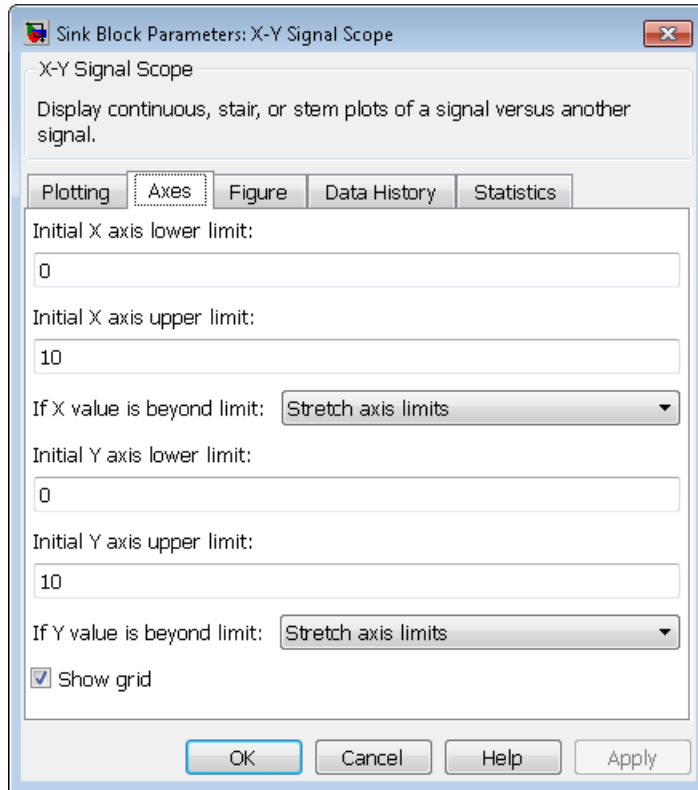
Plot type

The presentation format for the data. See “Connections Among Points in Plots” 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 “Vary Axis Limits Automatically”.

Initial Y axis lower limit, Initial Y axis upper limit

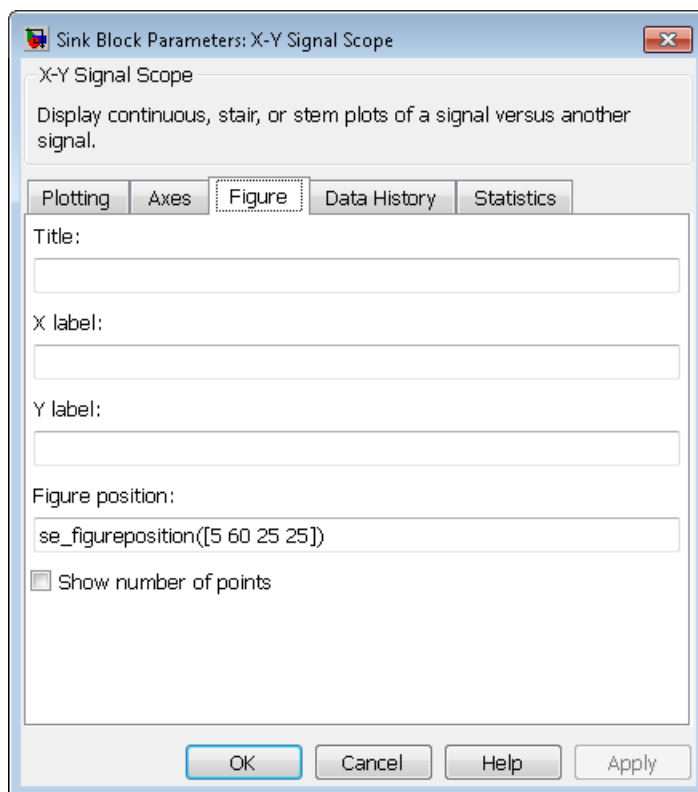
The interval shown on the Y axis at the beginning of the simulation. The interval might change from this initial setting due to zooming, autoscaling, or the **If Y value is beyond limit** setting.

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 “Vary Axis Limits Automatically”.

Show grid

Toggles the grid on and off.

Figure Tab**Title**

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

Y label

Text that appears to the left of the vertical axis.

X label

Text that appears below the horizontal axis.

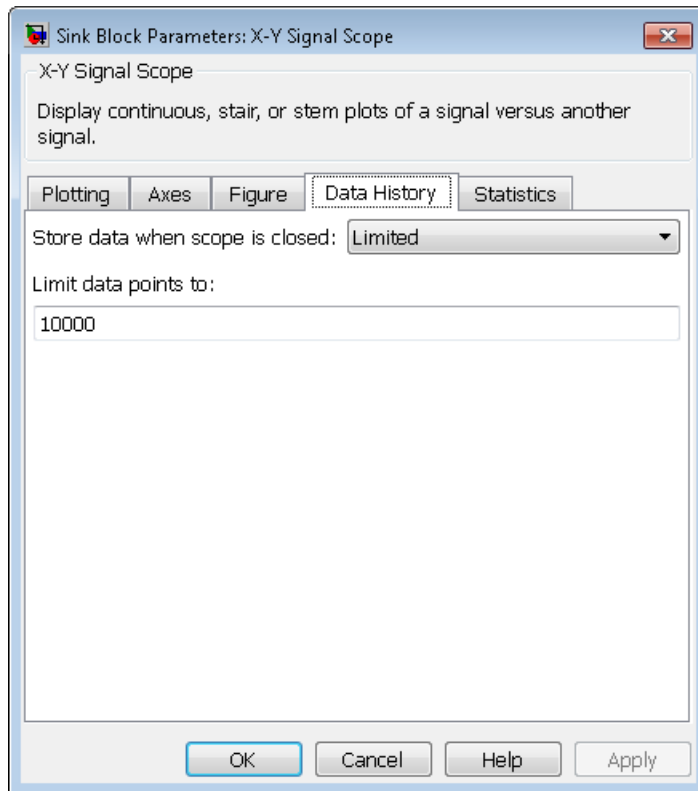
Figure Position

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

Show number of points

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

Data History Tab



Store data when scope is closed

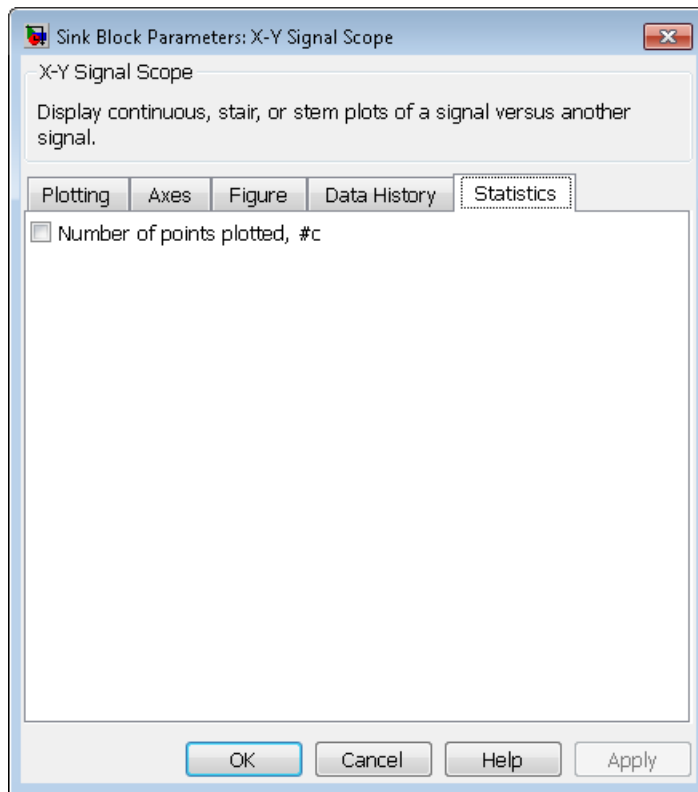
Select **Unlimited** to have the block cache all data for future viewing, **Limited** to cache a portion of the most recent data, and **Disabled** to avoid caching undisplayed data.

Limit data points to

The number of data points the block caches, using the most recent data. You see this field only if you set **Store data when scope is closed** to **Limited**.

Statistics Tab

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

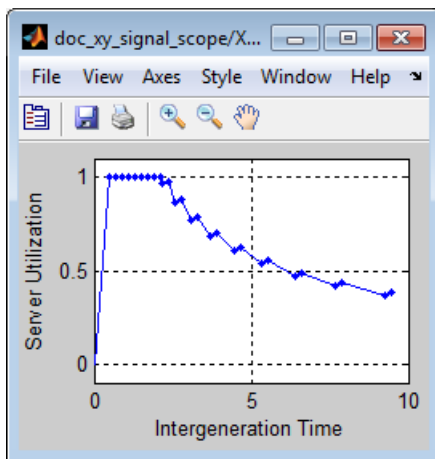
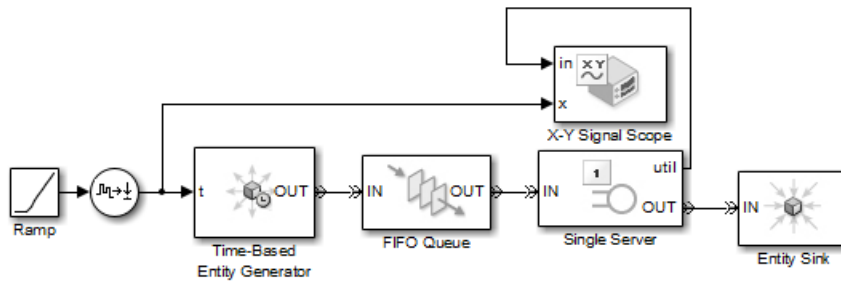


Number of points plotted, #c

Allows you to use the signal output port labeled #c.

Examples

The model below shows the relationship between the utilization of a server and the interarrival time of entities.



See Also

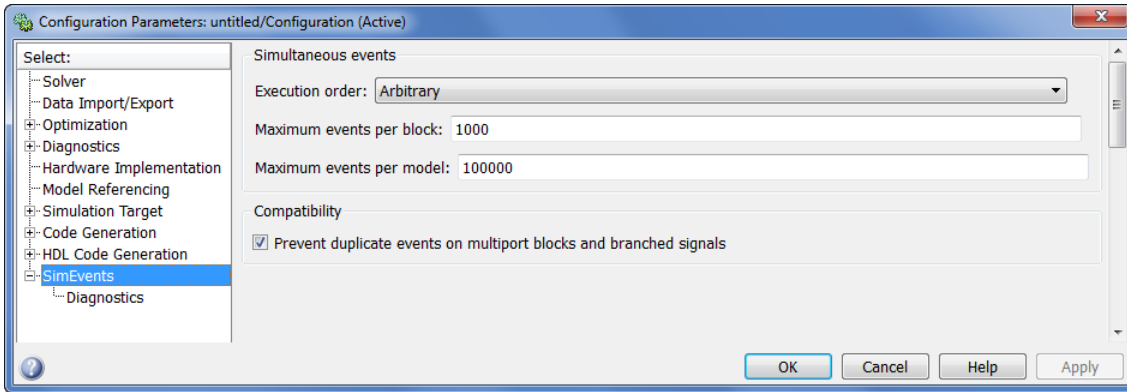
Signal Scope, X-Y Attribute Scope

“Choose and Configure Plotting Blocks”

Configuration Parameters

- “SimEvents Pane” on page 3-2
- “SimEvents Diagnostics Pane” on page 3-10

SimEvents Pane



In this section...

“SimEvents Pane Overview” on page 3-4

“Execution order” on page 3-5

“Seed for event randomization” on page 3-6

“Maximum events per block” on page 3-7

“Maximum events per model” on page 3-8

“Prevent duplicate events on multiport blocks and branched signals” on page 3-8

SimEvents Pane Overview

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. This is the maximum number of events per discrete-event system in a model.

Settings

Default: 100000

Minimum: 2

Maximum: $2^{31} - 1$

Command-Line Information

Parameter: propMaxDesMdlSimulEvents

Type: string

Value:

Default: '100000'

See Also

“Livelock Prevention”

Prevent duplicate events on multiport blocks and branched signals

Prevent multifiring behavior on multiport blocks or branched signals that results in duplication of events. Multifiring behavior, an implicit result of the way that the software executes particular block configurations, occurs when the software executes a block more than once in response to a single, discrete event in the simulation.

Settings

Default: On

On

Enable **Prevent duplicate events on multiport blocks and branched signals** parameter to prevent multifiring behavior.

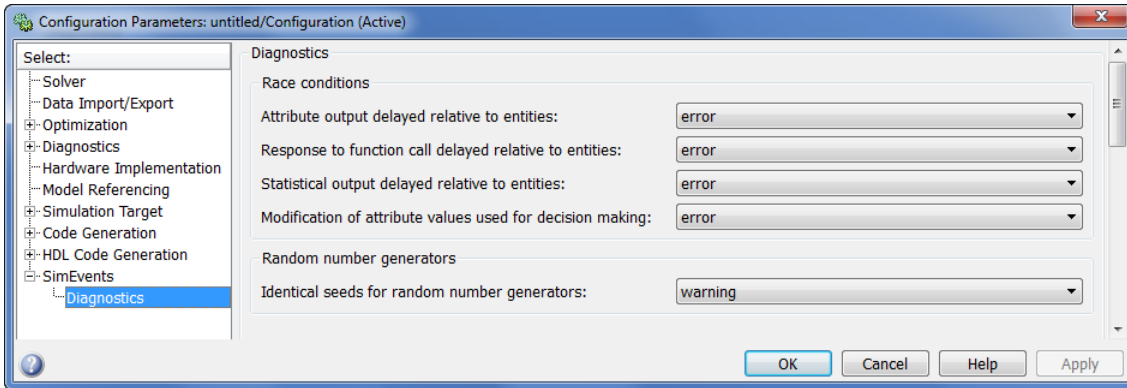
Off

Allow multifiring behavior on multiport blocks or branched signals.

Command-Line Information**Parameter:** propPreventDuplicateEvents**Type:** integer or boolean**Value:****Default:** '1' for integer, 'True' for boolean**See Also**

“Legacy Behavior in SimEvents Models”

SimEvents Diagnostics Pane



In this section...

“Diagnostics Pane Overview” on page 3-12

“Attribute output delayed relative to entities” on page 3-13

“Response to function call delayed relative to entities” on page 3-15

“Statistical output delayed relative to entities” on page 3-17

“Modification of attribute values used for decision making” on page 3-19

“Identical seeds for random number generators” on page 3-21

Diagnostics Pane Overview

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: error

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

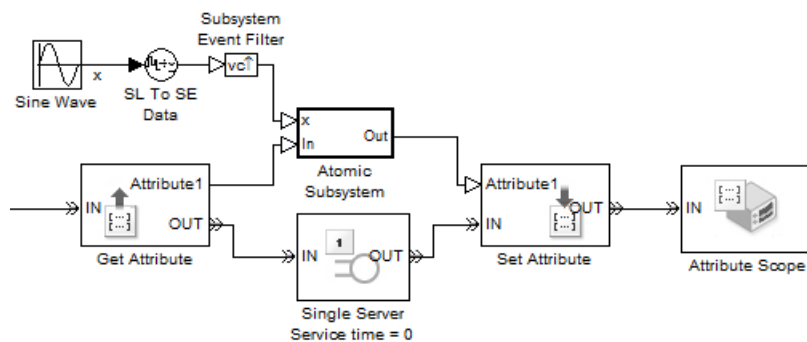
error

When the application detects this situation, it terminates the simulation and displays an error message.

Tip

A Single Server block whose **Service time** parameter is 0 can address the problem by storing the entity while the latter block responds to the signal update.

Example of Solution



For details, see “Use Block Diagrams to Manipulate Attributes”.

Alternatively, you might be able to address the problem by using an attribute directly instead of by using the signal output of a Get Attribute block. For an example, see “Use a Signal or an Attribute”.

Command-Line Information

Parameter: propDiagAttribOutput

Type: double

Value: 0 | 1 | 2

Default: 2

Recommended Settings

Application	Setting
Debugging	warning or error
Efficiency	none

See Also

- “Unexpected Use of Old Value of Signal”
- Fine-Grained Management of Simultaneous Events example

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: error

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

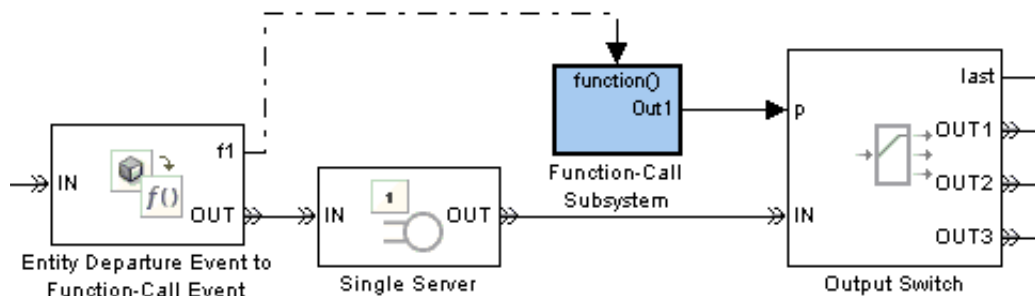
error

When the application detects this situation, it terminates the simulation and displays an error message.

Tip

A Single Server block whose **Service time** parameter is 0 can address the problem by storing the entity while subsequent blocks respond to the function call and its consequences.

Example of Solution



Command-Line Information

Parameter: propDiagFcnCallOutput

Type: double

Value: 0 | 1 | 2

Default: 2

Recommended Settings

Application	Setting
Debugging	warning or error
Efficiency	none

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: error

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

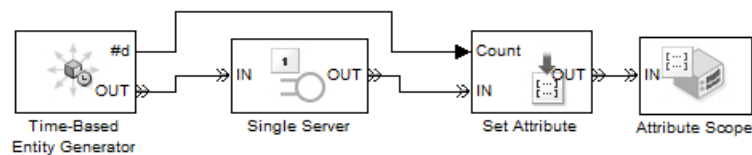
error

When the application detects this situation, it terminates the simulation and displays an error message.

Tip

A Single Server block whose **Service time** parameter is 0 can address the problem by storing the entity while the latter block responds to the signal update.

Example of Solution



For details, see “Sequence of Departures and Statistical Updates”.

Command-Line Information

Parameter: propDiagStatOutput

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”

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: error

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

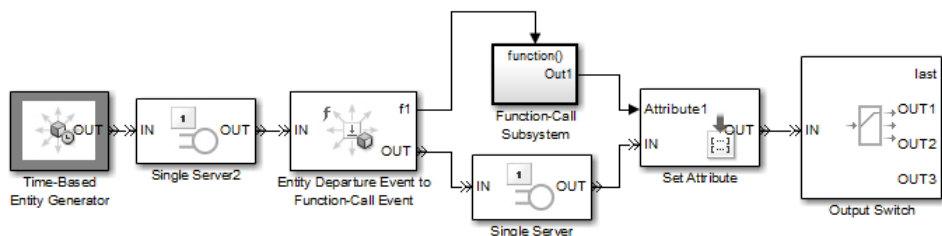
error

When the application detects this situation, it terminates the simulation and displays an error message.

Tip

A Single Server block whose **Service time** parameter is 0 can address the problem by storing the entity while the latter block responds to the signal update.

Example of Solution



Command-Line Information

Parameter: propDiagChangeAttrib

Type: double

Value: 0 | 1 | 2

Default: 2

Recommended Settings

Application	Setting
Debugging	warning or error
Efficiency	none

See Also

“Query 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_randomize_seeds` 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 Comparing Routing Policies example 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

Application	Setting
Efficiency	none

See Also

- [Managing Seeds During Random Number Generation example](#)
- [“Unexpected Correlation of Random Processes”](#)
- `se_randomize_seeds`

Upgrade Advisor Checks

SimEvents Upgrade Advisor Checks

In this section...
“Checks Overview” on page 4-2
“Check model and local libraries for legacy SimEvents blocks” on page 4-3
“Check for implicit event duplication caused by SimEvents blocks” on page 4-4

Checks Overview

Use SimEvents Upgrade Advisor checks to identify backward-compatibility issues in your model.

See Also

- “Check Model for Legacy Behavior”

Check model and local libraries for legacy SimEvents blocks

Check model and its associated libraries for blocks from a version of SimEvents prior to 4.0 (R2011b).

Description

This Upgrade Advisor check identifies blocks in your model from a version of SimEvents prior to 4.0 (R2011b).

Blocks from versions of SimEvents prior to 4.0 (R2011b) are called *legacy blocks*. If your model contains only legacy blocks, it continues to function as originally designed, but does not have the benefit of feature updates and modeling syntax changes available in the latest version of SimEvents. Your model cannot contain a *mix* of legacy blocks and blocks from version 4.0 (R2011b) or later of SimEvents because these versions are incompatible.

Available with SimEvents.

Results and Recommended Actions

Condition	Recommended Action
This model contains blocks from a version of SimEvents prior to 4.0 (R2011b).	Use the function <code>seupdate</code> to migrate the model to the latest version of SimEvents. For more information, refer to “Migration Using <code>seupdate</code> ” in the SimEvents documentation.

See Also

- “Visual Appearance of Legacy SimEvents Blocks”
- “Migration Using `seupdate`”

Check for implicit event duplication caused by SimEvents blocks

Check configuration parameters of model for status of **Prevent duplicate events on multiport blocks and branched signals** option.

Description

This Upgrade Advisor check verifies if you have selected the **Prevent duplicate events on multiport blocks and branched signals** check box in the Configuration Parameters dialog box of your model.

When you run a model created in a version of SimEvents prior to R2012a, the model might exhibit a behavior called *multifiring* that leads to duplication of events in the simulation. This event duplication behavior is implicit in models with certain configurations and results from the way the software executes the blocks of such configurations. Implicit event duplication is resolved in R2012a with the addition of the configuration parameter **Prevent duplicate events on multiport blocks and branched signals**.

Available with SimEvents.

Results and Recommended Actions

Condition	Recommended Action
SimEvents > Prevent duplicate events on multiport blocks and branched signals check box is not selected.	In the Configuration Parameters dialog box of your model, select the SimEvents > Prevent duplicate events on multiport blocks and branched signals check box.

An alternative to the recommended action in the preceding table is to use the **Modify Settings** button in the **Action** section of the Upgrade Advisor results pane. If you click **Modify Settings**, the software directly enables **Prevent duplicate events on multiport blocks and branched signals**.

Note: The configuration parameter **Prevent duplicate events on multiport blocks and branched signals** is not compatible with blocks from versions of SimEvents prior to 4.0 (R2011b). When you use the Upgrade Advisor to check your model for implicit event duplication, it first runs the check, “Check model and local libraries for legacy SimEvents blocks” on page 4-3. The Upgrade Advisor first provides the recommended action relating to the results of that check, before providing the recommended action (if

any) for the check, “Check for implicit event duplication caused by SimEvents blocks” on page 4-4.

See Also

- “Legacy Behavior in SimEvents Models”
- “Prevent duplicate events on multiport blocks and branched signals”

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 system

A system in which state transitions depend on asynchronous discrete incidents called events. You typically construct a discrete-event system by adding a variety of blocks, such as generators, queues, and servers, from the SimEvents block library.

One or more discrete-event systems can coexist with time-based systems in a Simulink model. Using special gateway blocks, you can pass signals from time-based components/systems to and from discrete-event components/systems modeled with SimEvents blocks.

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.

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.
gateway block	A block to convert between time-based and event-based signals or function-calls.
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	<p>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.</p> <p>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.</p>
pending entity	An entity that has tried and failed to depart from the block in which the entity resides. The failure occurs because the entity output port through which the entity would depart is connected to an unavailable entity input port of another block.
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	<p>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 $0 \leq \Delta t \leq 128 * \text{eps} * T$, where eps is the floating-point relative accuracy in MATLAB software and T is the simulation time.</p> <p>For example, in a D/D/1 queuing system where the arrival rate equals the service rate, an entity generation event and a service completion event are simultaneous. Parameters in the model determine which of these events occurs first, though the clock has the same value in both cases.</p>
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 “Role of Timeouts in SimEvents Models” and “Use Timeouts to Limit Entity Queueing Time” 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 “Role of Timeouts in SimEvents Models” and “Use Timeouts to Limit Entity Queueing Time” 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 “Multivalued Signals” for details.

