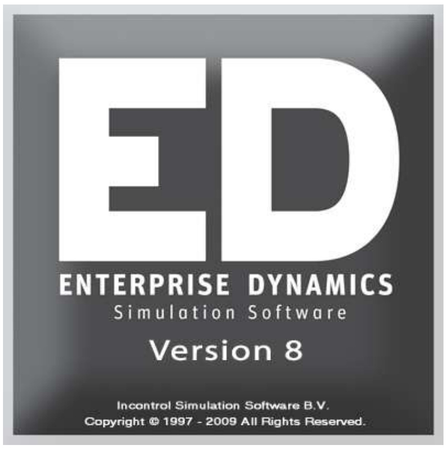
Description of the ED library
Basic Atoms


ED
ENTERPRISE DYNAMICS
Simulation Software
Version 8
Incontrol Simulation Software B.V.
Copyright © 1997 - 2009 All Rights Reserved.

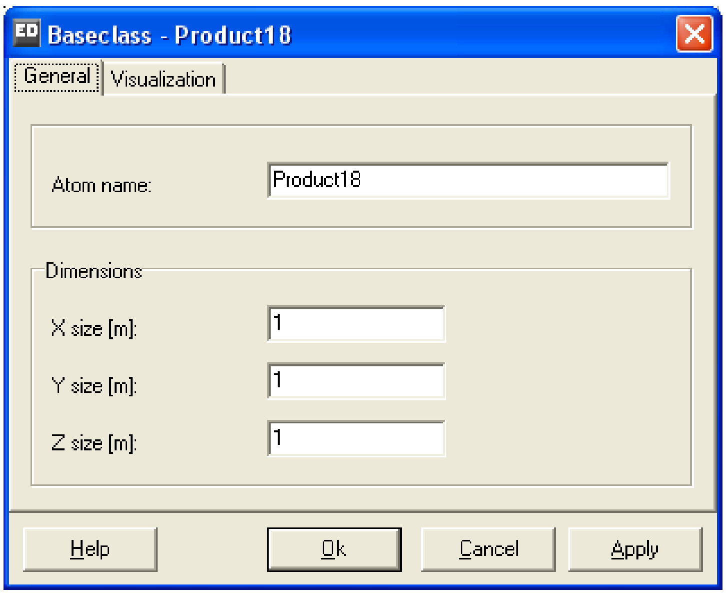
Simulation Software / Description of the ED library
BASIC ATOMS

INCONTROL
Simulation Solutions

Enterprise Dynamics®

Description of a few atoms

1 THE PRODUCT ATOM



Atom name: Product18

Dimensions

X size [m]: 1

Y size [m]: 1

Z size [m]: 1

Picture 1-1: The Product Atom

The Product Atom is used to model the physical flows in Enterprise Dynamics. These flows can consist of products, goods, documents or persons. The following atom settings can be defined:

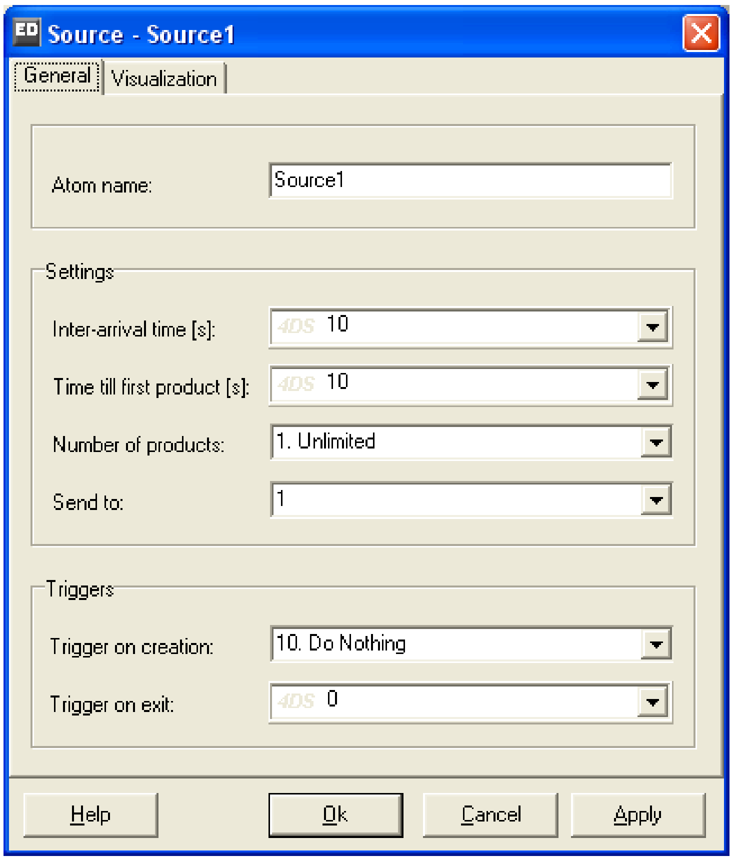
General Tab:

- *Atom name*
  The name given to the atom.
- *Size X*
  The size of the atom in the x direction (length in meters).
- *Size Y*
  The size of the atom in the y direction (width in meters).
- *Size Z*
  The size of the atom in the z direction (height in meters).

General visualization:

- *2D Icon*
  The symbol used to represent the Product atom in the 2D window.
- *Show 2D Icon*
  Gives you the possibility to display the 2D Icon. If the option is checked (standard setting), the icon is displayed.
- *3D Icon*
  The symbol used to represent the Product atom in the 3D window.
- *Color*
  The color given to the atom.

2 THE SOURCE ATOM


Source - Source1

General | Visualization

Atom name: Source1

Settings

Inter-arrival time [s]: 4DS 10

Time till first product [s]: 4DS 10

Number of products: 1. Unlimited

Send to: 1

Triggers

Trigger on creation: 10. Do Nothing

Trigger on exit: 4DS 0

Picture 2-1: The Source Atom

The Source Atom allows atoms, mostly products, to enter the model at a specified rate and therefore functions as a product or customer generator. This atom is often the first element of a model.

The following settings can be adjusted:

General Tab:

- *Atom name*
  The name given to the atom.
- *Inter-arrival time*
  Time between 2 product atom arrivals. This time is measured in seconds and can be constant, but also defined by a probability distribution. Click on the right triangle to display the pull down menu featuring a number of possible probability distributions with parameters and values.
- *Time till first product*
  Arrival time of the first product. After this first arrival the probability distribution from the inter-arrival time applies.

Default value is 10 seconds; if you want all the products have the same inter-arrival time, choose the same expression as used in the inter-arrival time.

- *Number of products*
  With this option it is possible to limit the entry of products to your model. There are two options:
  1. Unlimited (default)
  2. Generate maximum 100 products

  With option two you can choose your preferred number of arrivals. This is very similar to the more general Lock atom.
- *Send to*
  Displays the number of the output channel through which other atoms (mostly products) leave this atom. A number between 1 and the total number of the atom's output channels has to be displayed here. If the result is 0, sending never takes place. If an atom is blocked because the input channels of the receiving channels are closed, the 'Send to' statement is re-evaluated only when the situation changes and sending is allowed.

In the Send to option, the user can thus enter a figure or select one of the following pre-defined options. In these options, the **bold** items (shown in blue on the screen) can be altered by the user:

1: *Specific channel: always send to channel **1**.*
The Product Atom will always be sent to a defined output channel.

2: *An open channel (First channel first): search, starting from the first channel, and send to the first open channel found.*
The Product Atom is sent to the first open channel that Enterprise Dynamics finds. The search starts from the first output channel, then to number two and so on.

3: *An open channel (Last channel first): search, starting from the last channel, and send to the first open channel found.*
The product is sent to the first open channel Enterprise Dynamics comes across, starting from last channel to the one before and so forth.

4: *A random open channel: choose a random channel from all the open output channels.*
Enterprise Dynamics selects a random channel from all open channels. With long simulation runs, it results in equal utilizations of e.g. a group of servers.

5: *By percentage: **90%** of products go to channel **1**, the remaining percentage go to channel **2**.*
A definite percentage of the products is sent to a specific channel and the rest to another channel. The user can define the channels and the percentage.

6: *By atom name: if the atom name of the **1st** atom in the queue matches **AtomName** then send to channel **1** else **2**.*
The atoms are forwarded on the basis of their names. If the name corresponds to the name the user entered, the products are sent to channel **1** and otherwise to channel **2**. The user can adjust the channel numbers and the atom names.

7: *By label value (direct): the channel number is written directly on the label named **LabelName** of the **1st** atom in the queue. If the label value is 0 then send to channel **1**.*
The atoms are forwarded on the basis of a label value. The user has defined a

name for the label. The value of the variable corresponds to the output channel’s value. If the value is 0, a pre-defined exit is used. Note that searching for a label not present on the atom results in the value 0 as well.

8: *By label value (conditional): if the value on the label named **LabelName** of the **1st** atom in the queue is < the value **1** then send to channel **1** else **2**.*
Here too, the value of a specified label determines the choice of the output channel. If the value of the atom is lower than 1, the atom is sent to channel **1**, otherwise to channel **2**. All values and comparisons (lower than, higher than, equal to) can be edited.

9: *By label text: if the text on the label named **LabelName** of the **1st** atom in the queue matches **text** then send to channel **1** else **2**.*
When the value of a defined label is equal to a specific text, the atom is sent to channel 1, otherwise to channel 2. The text and the channel numbers are editable.

10: *Conditional statement: If **1** is > than **0** then send to channel **1** else send to channel **2**.*
If a specific value is higher than another value, the atom is sent to channel **1**, otherwise to channel **2**. The comparisons and channel numbers can be edited.

11: *By icon name: if the icon name of the **1st** atom in the queue matches **IconName** then send to channel **1** else **2**.*
If the name of the atom icon corresponds to a defined name, the atom is sent to channel **1**, otherwise to channel **2**. The icon names and channel numbers can be specified.

12: *By icon number: if the icon number of the **1st** atom in the queue is = the value **1** then send to channel **1** else **2**.*
If the atom’s icon number **is equal to 1**, the atom is sent to channel **1**, otherwise to channel **2**. The comparisons and channel numbers can be defined.

13: *Round robin: all output channels are used in rotation. If channel is closed, then wait till open.*
All output channels are used consecutively. If a channel is not open, Enterprise Dynamics waits until it becomes open.

14: *Lowest queue: Send to the channel connected to the atom with the lowest queue.*
The atom is sent to the output channel with the shortest queue. In the case of equal queues, the output channel with the lowest number is chosen.

15: *Largest queue: Send to the channel connected to the atom with the largest queue.*
The atom is sent to the output channel with the longest queue. In the case of equal queues, the output channel with the highest number is chosen.

16: *Lookup table: Send to the channel specified in row **1** column **2** of global table named **table1**.*
Sends the atom to the channel defined in row **1** and column **2** of a table. The row and column numbers, as well as the table name, can be specified.
Note that the table must be present in the model as a separate atom!

17: *Round robin if available: all output channels are used in rotation if channel is available. If channel is closed, then next available channel is chosen.*
All channels are used consecutively, but when the channel required is closed, the next available channel is selected.

18: *Matching icon number or empty: Sends to a queue containing products of same icon. If no icons match, then sends to first empty queue starting with last output channel.*
Forwards atoms so that they always arrive in a queue containing atoms with the same icon number. If a queue containing atoms with the same icon number is not found, ED searches for the first empty queue, starting from the queue connected to the last output channel.

19: *Lowest queue of next two atoms: Sends to the output channel connected to the lowest queue, where lowest queue takes into account the next TWO atoms.*
Enterprise Dynamics evaluates the total queue for each atom connected to an output channel, and the atom connected to that atom. It then sends the next atom to the channel connected to the queue with the lowest contents. For example, an atom can be sent to 3 different queues, each of which is followed by a server. This option prevents the products from being sent to a queue where the server is already in action, while the other servers are available.

20: *By user: enter your own 4DScript expression resulting in a value between 1 and the number of channels: 1. You can press the small button for the 4DScript editor.*
The user writes a 4DScript code that results in the output channel. Clicking on the small square button by the text will open the 4DScript editor.

21: *Random channel: randomly choose a channel. If the channel is open then send to it, otherwise choose again when any channel opens.*
Enterprise Dynamics chooses a random channel. If this channel is open, the product is sent to that channel. However, if it is closed, choose again when another channel opens.

- *Trigger on Creation*
The command in this field is performed when an atom enters the model. The user can define their own 4DScript expression, or pick from one of the following options:

1: *Assign label: products are assigned a label named* ***LabelName*** *with a value of* ***1****.*
The products are assigned a label with a specific name and a definite value. The label name and the value can be defined.

2: *Auto Name: a counter is added to the end of each product's name.*
A counter is added to the product’s name. The first product is called for example Product1 and the second Product2.

3: *Random icons: products are assigned a random icon number between* ***2*** *and* ***6****.*
A random icon is allocated to each product. The icon number lies between two defined values.

4: *Set Size: product dimensions are set to: X=* ***50*** *cm, Y=* ***40*** *cm, Z=* ***30*** *cm.*
The product’s dimensions change according to the entered values.

5: *Random Size: product dimensions are randomly set within the following ranges: X=* ***50*** *to* ***100*** *cm, Y=* ***50*** *to* ***100*** *cm, Z=* ***50*** *to* ***100*** *cm.*
The product’s dimensions change according to random values inside defined ranges.

6: *Set Color: products are set to the **colorpurple**.*
A product's color changes into the color defined by the user. Note that in 4DScript, the selected color has to be prefixed by the word "color". So *colorpurple* is the command for purple. Instead of the command *colorpurple*, you can also enter the color number.
7: *Random color: products are assigned a random color.*
The products are given a random color.
8: *Random Size and Color: products are assigned a random color and its dimensions are randomly set within the following ranges: X=**50** to **100** cm, Y= **50** to **100** cm, Z= **50** to **100** cm.*
The products get a random color as well as a random size (within defined ranges).
9: *Outline: display the products as a simple outline, not its icon.*
A product's icon is not visible any more, only its outline is.
10: *Do Nothing.*
Nothing happens. This is the standard setting.

- *Trigger on Exit*
The command in this field is executed when a product is leaving the atom. You can either use your own 4DScript command, or one of the pre-defined expressions. The question marks indicate where the user must enter a value.

The possible pre-defined expressions in the Trigger on Exit field are:
1: *setlabel([**?**],**?**,i).*
With this 4DScript command, a label is added to the atom leaving the Source. The code is: setlabel([**label name**], **value**, i).

Example
To allocate a label "complete" with the value 1 to the product, use the following code: setlabel([complete], 1, i).
The letter i refers to the *involved* atom. This is the atom undergoing the process of leaving the Source. If a label had to be placed on the Source itself, the i could be replaced by a c (of *current*).

Example
Setlabel([cycletime], Uniform(25,45), i) defines a label "cycletime" on the product with a value from a uniform distribution of between 25 and 45 seconds. This result can later be used as cycle time for a server, for instance.

2: *set(Name(i),[**?**]).*
Changes the name of the atom leaving the Source. The '?' must be replaced by the name chosen for the atom.
3: *set(Icon(i),**?**).*
Changes the atom's icon into the icon with the number '?'.
4: *set(Icon(i), IconByName([**?**])).*
Changes the icon into the icon with the name '?'.
5: *set(Color(i), **ColorYellow**).*
Changes the atom's color into the defined color. In Enterprise Dynamics, the colors must be specified either by their number or with the prefix "color" followed by the color in question, for example ColorRed.

6: *setsize(**?**,**?**,**?**,i).*
Changes the atom's dimensions according to the defined sizes (x,y,z).

7: *setloc(**?**,**?**,**?**,i).*
Gives the atom a new location, as defined in the command (x,y,z).

8: *FreeOperators(AtomByName([**Team**], Model), i).*
Allows the Operator atoms to be re-used. Replace 'Team' by the name of the Team atom. This option is only for advanced users.

9: *if(=(**?**,**?**),**?**,**?**).*
A conditional comparison. For example, entering the following code gives:

```
If(=(thesis1,thesis2),command1,command2)
```

If thesis1 and thesis2 are equal, command1 will be executed, otherwise command2. Command2 can also be omitted.

10: *if(=(label([**?**],i),**?**),**?**,**?**).*
A conditional comparison, in which a label's value is considered.

Example
With the following command, if the label 'Reject' has the value 1, we can color all rejected products red and all approved products green:
```
if(Label([Reject], i) = 1, Color(i) := ColorRed, Color(i), ColorGreen).
```

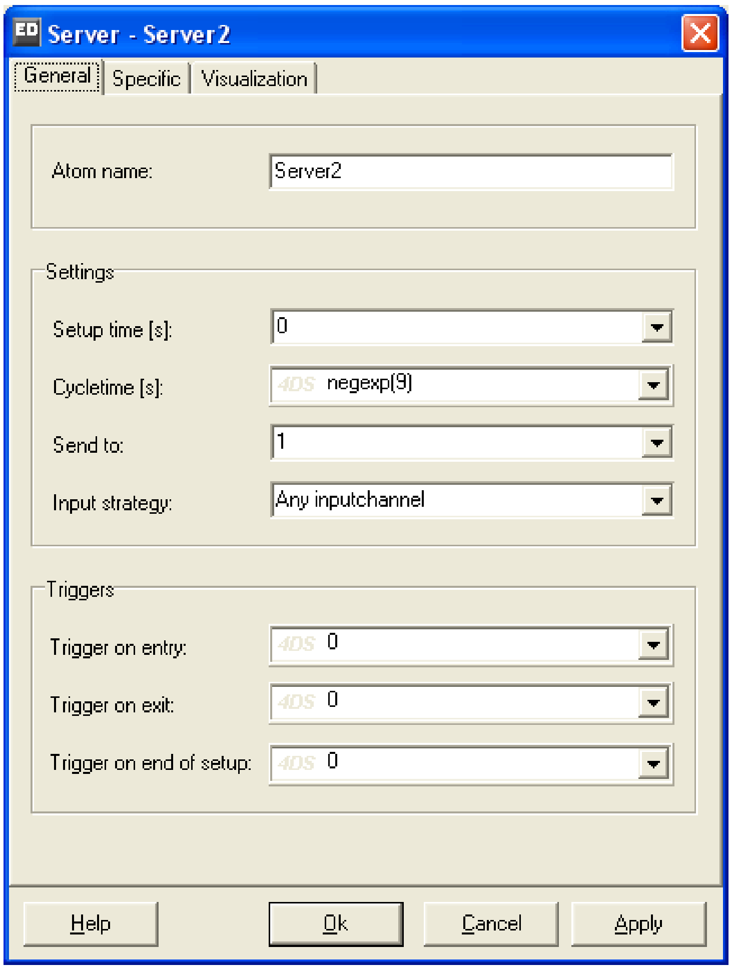
11: *if(CompareText(Name(i),[**?**]),**?**,**?**).*
A conditional comparison, in which the atom's name is used. Its functioning is otherwise the same as option 10.

Vizualization tab:

- *Icon*
  The symbol used to represent the Source atom in the 2D window.

For a more detailed explanation relating to 4DScript commands, we refer you to Annex 3 or the help files included in Enterprise Dynamics.

3 THE SERVER ATOM


Server - Server2

General | Specific | Visualization

Atom name: Server2

Settings

Setup time [s]: 0

Cycletime [s]: 4DS negexp(9)

Send to: 1

Input strategy: Any inputchannel

Triggers

Trigger on entry: 4DS 0

Trigger on exit: 4DS 0

Trigger on end of setup: 4DS 0

Picture 3-1: The Server Atom

The Server is used to model operations taking a certain amount of time such as the processing of a product by a machine or a customer's settlement at a cash desk. As a result, the Server can represent a machine, a counter, an assistant or another type of processing place or device. As well as cycle times, other parameters can be defined such as setup times or the simultaneous processing of several products.

The following values are editable:

General tab:

- *Atom name*
  The name given to the atom.

- *Setup time*
  Time needed before the actual processing starts. For example: cleaning of machines, adjusting settings for new products, etc.

Clicking on the triangle opens a series of options, including one that allows the settings to be defined for each product, or for products of a different type only. In addition to using these options, the user can also create his own 4DScript.

- *Cycletime*
  The time needed to process the product. By clicking on the arrow, a list of pre-defined 4DScript commands appears.
  Important: in the case of a grouped processing of products (batch processing), the cycle time refers to the whole batch and not to each individual product. Further, the processing starts only when the batch is complete.

- *Send to*
  Displays the channel to which the products have to be sent.
  For more details: see the ‘Send to’ explanation of the Source atom.

- *Input strategy*
  Regulates the access to an atom from previous atoms via their output channels to this specific atom. The input strategy has several roles: it can open one or more channels and it can define the order in which products will be accepted from the available channels.

  You can compare input strategy to the sequencing of traffic lights, where some traffic lights are switched from ‘red’ to ‘green’ for one or several minor roads, irrespective of the actual traffic, and where the processing priority of these minor roads is defined.

  Warning: the first three input strategies open all input channels and the last two open one input channel each time!

1: *Any inputchannel.*
When activated, this strategy opens all input channels of an atom. If more than one of the atoms that are connected via the input channel can be sent, the atom arriving through lowest number input channel will have priority. While products keep entering through the first channel, the other channels will be blocked.

2: *Largest queue.*
When activated, this strategy opens all input channels of an atom. If more than one of the atoms that are connected via the input channel can send, the atom with the longest queue or largest contents will have priority. Note that in the case of several equally long queues, the input channel with the lowest number is chosen.

3: *Longest waiting.*
When activated, this strategy opens all input channels of an atom. If more than one of the atoms that are connected via the input channel can send, the atom with the highest average waiting time will have priority. In the case of several atoms with an equal waiting time, the input channel with the lowest number is always chosen. Note that it does not mean that the queues get approximately equally long, as is the case in the previous option.

4: *Round robin.*
This strategy first opens the first input channel and then waits for a product to be sent through this input channel. In the second cycle, it is the turn of the second input channel etc. When the products have run through the last input channel, the procedure is resumed with the first one.
Important remark: this strategy becomes active <u>after</u> the first product! So, in case of three input channels this strategy gives x,2,3, 1,2,3, 1,2,3 where x can be 1,2 or 3!

5: *Channel **1**.*
In this case, you can enter a specific input channel through which all products must enter. If 1 is entered, the products may only enter through input channel 1. Note that this rule does not apply to the first product entering as all channels are open initially.

- *Trigger on Entry*
  The command entered in this field is performed when a product enters the Server.
  For more details: see the 'Trigger on Exit' explanation of the Source.

- *Trigger on Exit*
  The command entered in this field is performed when a product leaves the Server.
  For more details: see the 'Trigger on Exit' explanation of the Source.

- *Trigger on end of setup*
  A rule that determines what kind of action needs to be executed at the end of the server setup time. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

Specific Tab:

- *Batch (B)*
  The batch size can be entered here. The standard setting is 1.

- *Batch rule*
  To set up the batches. There are 3 options:
  1: *B in, 1 out (the first).*
  As soon as the number of products that have entered the Server is equal to the batch size, the product at the front is forwarded to the next atom. The other products disappear.
  2: *B in, B out.*
  As soon as the number of products that have entered the Server is equal to the batch size, the products are forwarded to the next atom. The Server re-admits products only when all products of the batch have left the Server.
  3: *1 in, B out (copies of in).*
  Each time a product enters the Server, as many products as defined in the Batch input field leave the atom. The products are all copies of the atom that entered the Server. The Server re-admits a product only when all other products have left the atom.

- *Busy time*
  When this option is checked, the time taken into consideration in the "Mean Time Between Failure" options is only the time that the Server is actually in use, and not the total simulation time.

- *MTTF*
  This abbreviation stands for Mean Time To Failure, that is the average time elapsing between the end of a repair and the beginning of next failure. This average time between two Server failures can be defined in the input field. The time must be entered in seconds.

- *MTTR*
  This abbreviation stands for Mean Time To Repair. The average time needed to fix the Server can be defined in the input field.

- *MCBF*
  Abbreviation for Mean Cycles Between Failure. The average number of cycles between two failures can be entered in the input field. In MCBF, there is not a definite time between two failures but a definite number of batches.
  NB: When both fields MTBF and MCBF are filled in, failures will be generated by both definitions.

- *MTTR for cycles*
  This Mean Time To Repair input field applies to failures defined by MCBF.

- *Trigger on Breakdown*
  The command entered in this field is performed at the start of the Breakdown period of the Server.
  For more details: see the 'Trigger on Exit' explanation of the Source.

- *Trigger on Repair*
  The command entered in this field is performed at the start of the Repair period of the Server. For more details: see the 'Trigger on Exit' explanation of the Source.

Visualization Tab:

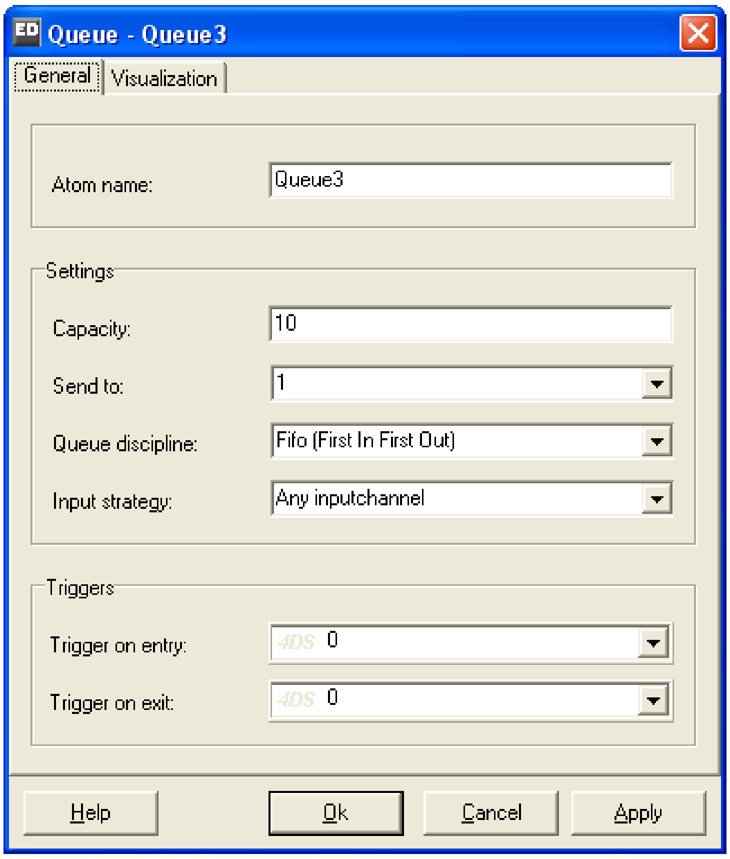
- *Icon*
  The symbol used to represent the Server atom in the 2D window.

- *3D Icon*
  The symbol used to represent the Server atom in the 3D window.

- *Main color*

- *Second color*

4 THE QUEUE ATOM



Atom name: Queue3

Settings

Capacity: 10

Send to: 1

Queue discipline: Fifo (First In First Out)

Input strategy: Any inputchannel

Triggers

Trigger on entry: 4DS 0

Trigger on exit: 4DS 0

Picture 4-1: The Queue Atom

When the next atom is occupied, the Queue atom places products in a queue. The following settings can be adjusted:

General tab:

- *Atom name*
  The name given to the atom.

- *Capacity*
  The Queue's capacity. When as many products are present in the queue as defined in the capacity input field, no new products can be placed in the queue.

- *Send to*
  Displays the output channel to which the products have to be sent.
  For more details: see the Send to explanation of the Source atom.

- *Queue discipline*
  The way products are arranged in the queue. The following options are possible:

1: *First in first out.*
The atoms are put in the queue according to their order of entry.
2: *Last in first out.*
The entering atoms are placed at the front of the queue. Consequently, the products leave the queue in reverse of their order of entry.
3: *Random.*
This queue discipline places the incoming products in a random spot in the queue
4: *Sort by **Label** Ascending.*
The products with the lowest value for a specific label are placed at the front of the queue.
Warning!: if the products are not sorted properly, the cause might be a space before or after the label name.
5: *Sort by **Label** Descending.*
The products with the highest value for a specific label are placed at the front of the queue.
Warning!: if the products are not sorted properly, the cause might be a space before or after the label name.
6: *User defined.*
The products are placed in the queue according to a position defined by the user.

- *Input Strategy*
  This input window can be used for indicating which input channel is to be used**.**
  For more details: see the 'Input Strategy' of the Server atom.

- *Trigger on Entry*
  The command entered in this field is performed when a product enters the Queue atom.
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

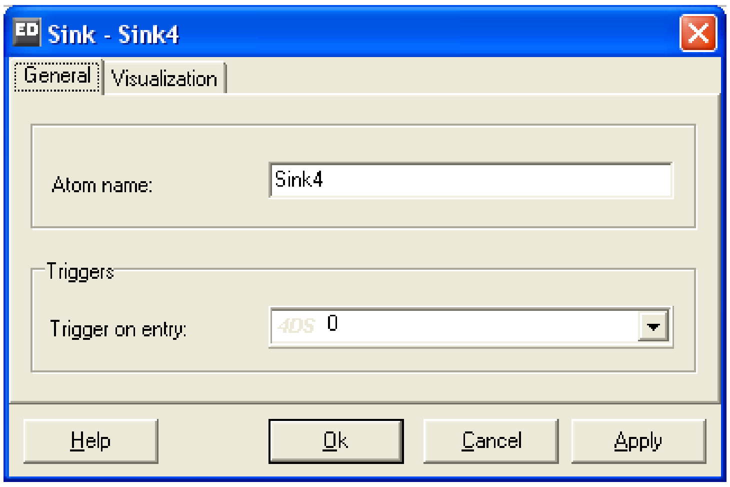
- *Trigger on Exit*
  The command entered in this field is performed when a product leaves the Queue atom.
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

Visualization tab:

- *Icon*
  The symbol used to represent the Queue atom in the 2D window.

- *3D Icon*
  The symbol used to represent the Queue atom in the 3D window.

5 THE SINK ATOM


Sink - Sink4

Atom name: Sink4

Triggers

Trigger on entry: 4DS 0

Picture 5-1: The Sink Atom

This atom allows products to leave the model. The following settings can be adjusted:

General Tab:

- *Atom name*
  The name given to the atom.

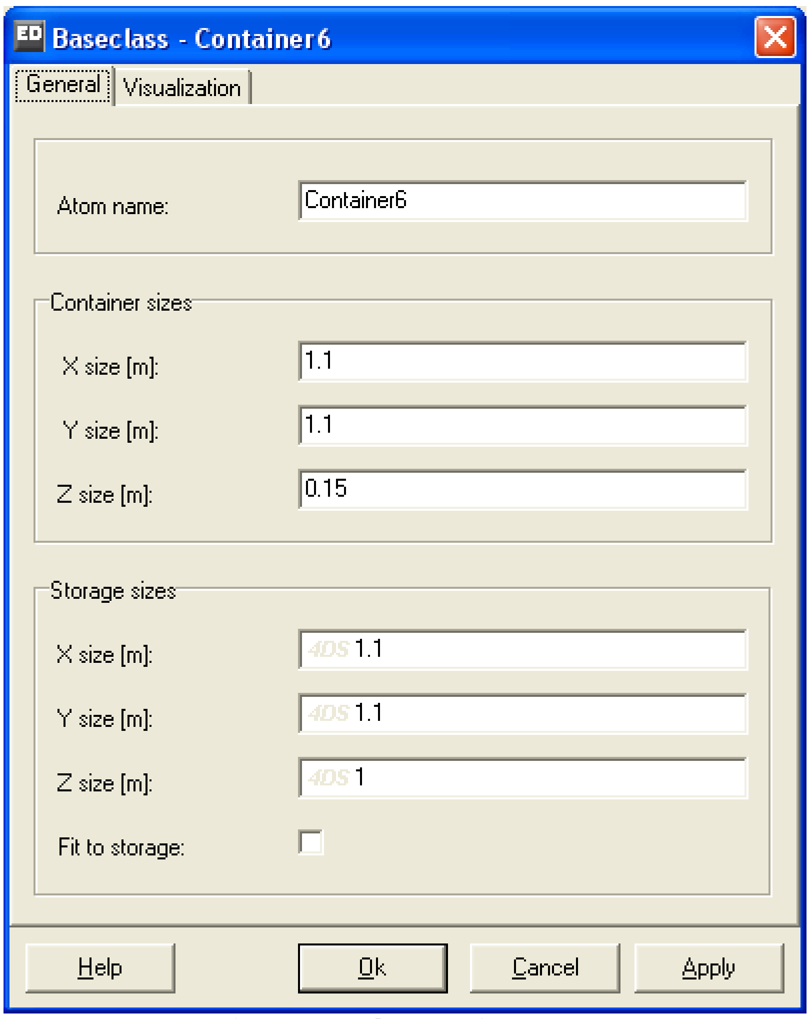
- *Trigger on Entry*
  The command entered in this field is executed as soon as a product enters the Sink.

Visualization Tab:

- *Icon*
  The symbol used to represent the Sink atom in the 2D window.

- 3D Icon
  The 3D icon that is used to represent the Sink atom in the 3D window.

6 THE CONTAINER ATOM


Baseclass - Container6

General | Visualization

Atom name: Container6

Container sizes

X size [m]: 1.1

Y size [m]: 1.1

Z size [m]: 0.15

Storage sizes

X size [m]: 4DS 1.1

Y size [m]: 4DS 1.1

Z size [m]: 4DS 1

Fit to storage: ☐

Picture 6-1: The Container Atom

The Container atom is created especially for storing or stacking other atoms, such as boxes or pallets. For the Container atom, a number of standard options, such as special 3D icons, have been designed for improved visualization. Moreover, the size of a product can automatically be adjusted to the size of the Container. In principle, the Container atom, like the Product atom, is placed in the model via a source (or via the Arrival list atom). Using an Assembler atom, products can then be put in the Container.

General Tab:

- *Atom name*
  The name given to the atom.

- *Container X size*
  The size of the container measured in the x direction (length).

- *Container Y size*
  The size of the container measured in the y direction (width).

- *Container Z size*
  The size of the container measured in the z direction (height).

- *Storage X size*
  Lists the size (in the x direction) required for storing an arriving product. Entering more space than actually required by the product results in empty spaces between the products.

- *Storage Y size*
  Lists the size (in the y direction) required for storing an arriving product. Entering more space than actually required by the product results in gaps between the individual products.

- *Storage Z size*
  Lists the size (in the z direction) required for storing an arriving product. Entering more space than actually required by the product (see Product atom) results in gaps between the products.

- *Fit to storage*
  Checking this option adjusts the size of a product such that it exactly matches the pre-defined storage size.

Warning!
There are 3 types of sizes: the size of the product (see Product atom), the size of the container and the storage size. The storage size allows (visual) adjusting of the packing method.

Example:
A pallet, Container with a Container size of 1x1x0.15, will stack a product sized 1x1x1 4 high if the Assembler places 4 products on a pallet.
But when Fit to Storage is selected with a storage size of 0.5x0.5x0.5, it will neatly place 4 boxes on one layer!

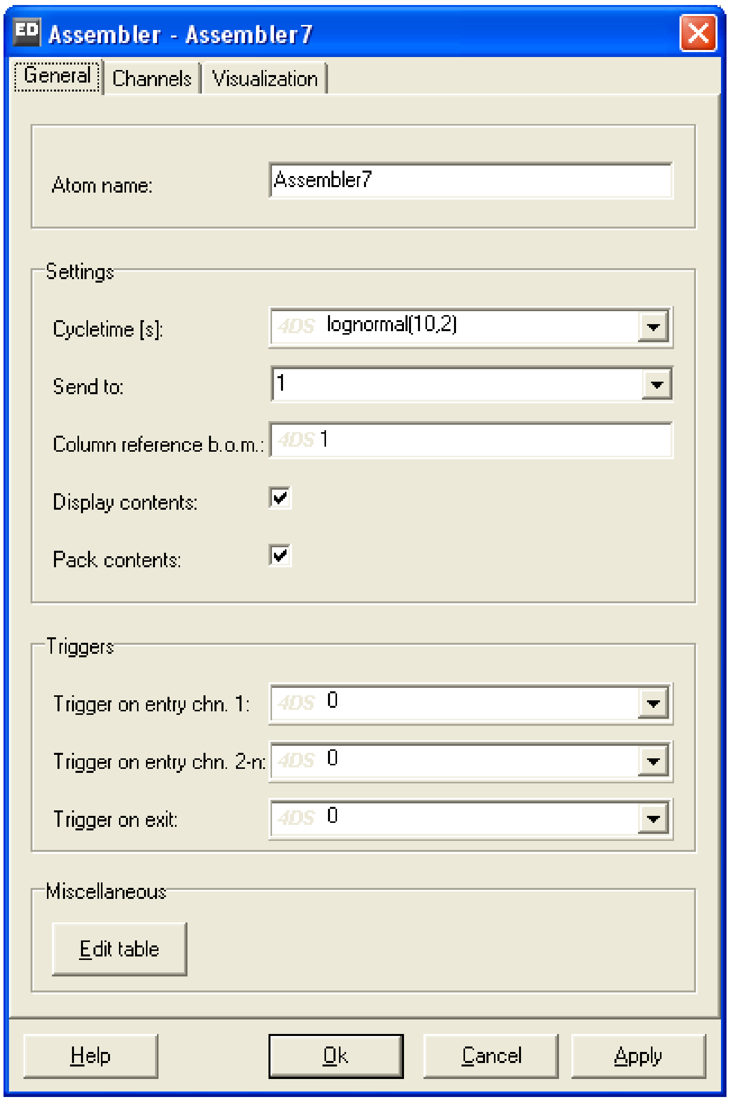
Visualization Tab:

- *Color*
  The color given to the atom.

- *2D Icon*
  The symbol used to represent the Container atom in the 2D window.

- *3D Icon*
  The symbol used to represent the Container atom in the 3D window.

7 THE ASSEMBLER ATOM


Assembler - Assembler7

General | Channels | Visualization

Atom name: Assembler7

Settings

Cycletime [s]: 4DS lognormal(10,2)

Send to: 1

Column reference b.o.m.: 4DS 1

Display contents: ✓

Pack contents: ✓

Triggers

Trigger on entry chn. 1: 4DS 0

Trigger on entry chn. 2-n: 4DS 0

Trigger on exit: 4DS 0

Miscellaneous

Picture 7-1: The Assembler atom

This atom merges atoms from several sources. Atoms placed into another atom can remain stored or be destroyed. Besides simulating real assembly work, this atom is also very useful for packing products in boxes or on pallets, and even for compiling orders.

The pallet, box or order always enters via the first input channel, while the products enter via the other channels. Depending on the settings, these products are destroyed or placed in the atom that has entered via the first input channel.

General tab:

- *Atom name*
  The name given to the atom.

- *Cycletime*
  The time needed to combine the products. This time is measured from the moment when all the atoms required have entered the Assembler, and refers to the whole assembly operation and <u>not</u> to each individual product! Clicking on the arrow opens a number of pre-defined 4DScript expressions.

- *Send to*
  Indicates to which exit channel the products are sent. For more details: see the 'Send to' explanation of the Source atom.

- *Column reference b.o.m.*
  Indicates what column from the b.o.m. will be used. The user can enter a figure, but also a 4DScript command that produces a figure. This field is evaluated when the first atom arrives via input channel 1 and defines which column will be used when the other atoms enter the Assembler.

  <u>Bill of Material</u>
  The Assembler atom also has a bill-of-material (b.o.m.) table, (visible by clicking the "edit Table" button). The b.o.m. lists, by input channel, how many atoms are required for assembling the end product.

  This b.o.m. has a number of rows and columns. The number of columns matches the number of input channels. Although the default setting is 1, the user can define the number of columns himself.

  This means that a separate column can be created for each product type that is assembled. When a product enters via the first input channel, the user defines which column in the b.o.m. is used for the rest of the arriving atoms.

- *Display contents*
  Checking this option displays the contents of the Assembler.

- *Pack contents*
  Checking this option, the atoms (read: products) are placed in the atom that entered via channel 1 (read: main product or Container). If this option is not checked, all atoms that did not enter via channel 1 will be destroyed.

- *Entry Trigger channel 1*
  The command entered in this field is performed when a product enters the Assembler via the first input channel.
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

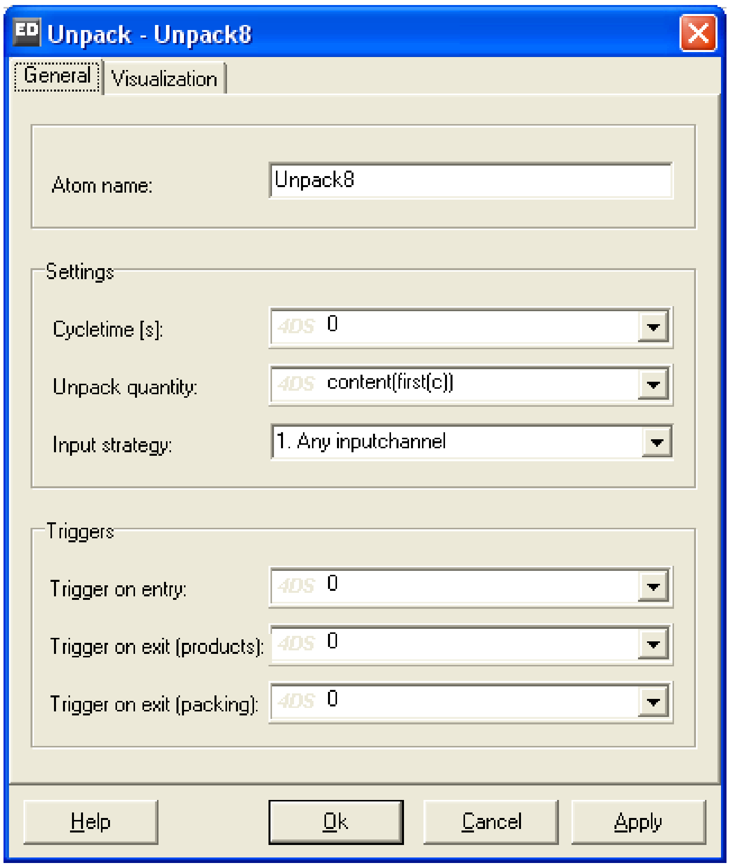
- *Entry Trigger channel 2..n*
  The command entered in this field is executed when a product enters the Assembler via one of the other input channels (i.e. not via channel 1).
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

- *Trigger on exit*
  The command in this field is executed when a product leaves the Assembler.
  For more details: see the ‘Trigger on Exit’ explanation of the Source atom.

Visualization tab:

- *Icon*
  The symbol that represents the Assembler atom in the 2D window.

8 THE UNPACK ATOM


Unpack - Unpack8

General | Visualization

Atom name: Unpack8

Settings

Cycletime [s]: 4DS 0

Unpack quantity: 4DS content(first(c))

Input strategy: 1. Any inputchannel

Triggers

Trigger on entry: 4DS 0

Trigger on exit (products): 4DS 0

Trigger on exit (packing): 4DS 0

Help | Ok | Cancel | Apply

Picture 8-1: The Unpack atom

The Unpack atom is used for removing products from a Container atom. After unpacking, the Container atom is sent via the second output channel and the products via the first output channel.

General tab:

- *Atom name*
  The name given to the atom.

- *Cycletime*
  The time needed to unpack the container. This time is measured from the moment the container enters the atom and refers to the 'unloading time' per container (and *not* to that of each individual product). Clicking on the arrow will open a number of pre-defined 4DScript expressions.
  For more details: see the 'Cycletime' explanation of the Server atom.

- *Unpack quantity*
  Defines the number of atoms that will be unpacked. When this number is

reached, the container is sent through. The default setting `content(first(c))` ensures that all products are removed from the container. The 3 other pre-defined options are:
duniform(1,10)
label([?],first(c))
if(=(?,?),?,?)

- *Input strategy*
  This field may be used for indicating which input channel should be used.
  For more details: see the 'Input strategy' of the Server atom.

- *Trigger on entry*
  The command entered in this field is performed when a container enters the Unpack atom.
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

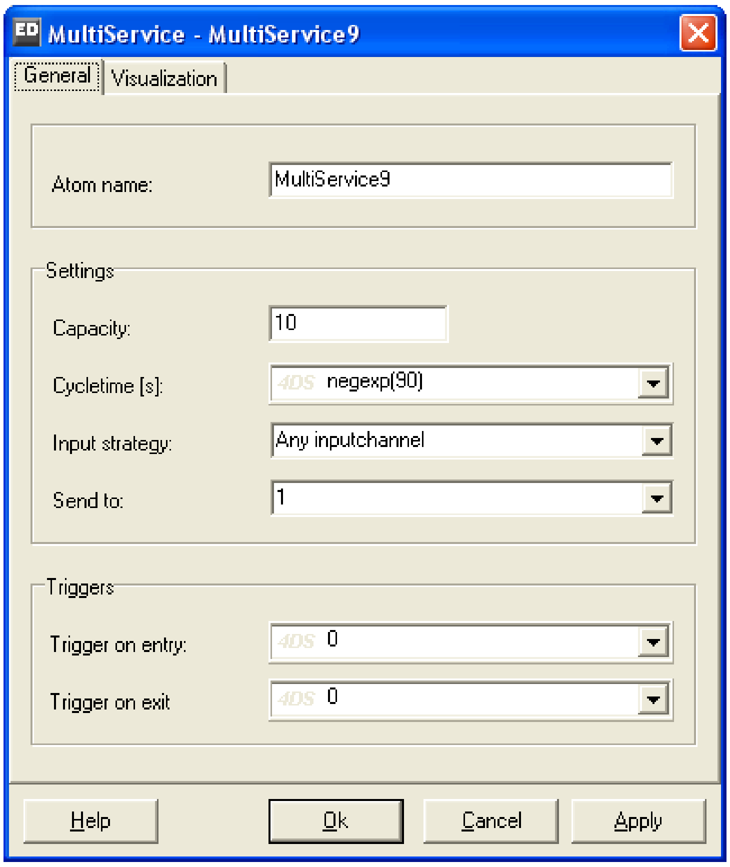
- *Exit trigger products*
  The command entered in this field is performed when a product leaves the Unpack atom.
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

- *Exit trigger packaging*
  The command entered in this field is performed when a container leaves the Unpack atom.
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

Visualization tab:

- *Icon*
  The symbol used to represent the Unpack atom in the 2D window.

9 THE MULTISERVICE ATOM



MultiService - MultiService9

General | Visualization

Atom name: MultiService9

Settings

Capacity: 10

Cycletime [s]: 4DS negexp(90)

Input strategy: Any inputchannel

Send to: 1

Triggers

Trigger on entry: 4DS 0

Trigger on exit 4DS 0

Picture 9-1: The MultiService atom

The MultiService atom has the same basic functions as several individual Server atoms. Where the Server atom can process only one product at a time, the MultiService atom allows the simultaneous processing of several products. The settings options are not as elaborate as those of the normal server, but all basic functions are available.

General tab:

- *Atom name*
  The name given to the atom.
- *Capacity*
  Lists the number of products that can be processed simultaneously.

For the options below, we refer you to the description of the Server atom:

- *Cycletime*
- *Input strategy*
- *Send to*
- *Trigger on Entry*

- *Trigger on Exit*

Visualization tab:

- *Icon*
  The symbol used to represent the MultiService atom in the 2D window.

- *2D display*
  The way in which the products are displayed in the atom in 2D:
  1: *Standard display*
  The products in the MultiService atom are not visible, but a text display indicates how many products are present.
  2: *Move left right*
  The products are visible and move from the left to the right.
  3: *Move right left*
  The products are visible and move from the right to the left.
  4: *Level vertical*
  Instead of the products, a colored area is visible which increases in height when more products enter the MultiService atom.
  5: *Level horizontal*
  Instead of the products, a colored area is visible which scrolls to the right when more products enter the MultiService atom.
  6: *Line up content*
  The products are visible and displayed one underneath the other.