ANNEXES
TUTORIAL ED6

Simulation Software

Planetenbaan 21
3606 AK Maarssen
The Netherlands
www.EnterpriseDynamics.com

Annex 1 The menu structure

NB: An option displayed in *italic* is more suitable for the advanced user.

File	Explanation
New Model	Closes the current model and opens an empty model.
Open Model…	Closes the current model and opens an existing model.
Merge Model…	*Opens a model and includes this model into the selected atom or into the current model. Consequently, the existing model is not deleted.*
Save Model	Saves the current model under the same file name.
Save Model As…	Saves the current model under a new file name. The previous model remains.
Add Atom to Library…	Selects an atom and places it at the bottom of the library.
Save Atom As…	*Displays a window allowing you to select an atom out of the library. Afterwards, the selected atom can be saved under a new name.*
Import	*This option allows you to import a 2D/VR (VR=Virtual Reality) icon or a VR sound. The icon or sound is added to the lists with icons and sounds that the user can allocate to an atom.*
Print 2D Layout…	Prints the 2D window on the standard printer.
Print Setup	Allows you to define the settings of the standard printer.
Preferences	This option displays a number of tabs to define the standard settings of Enterprise Dynamics.
Startup Script	*Allows you to modify the Startup Script. This script is performed each time Enterprise Dynamics is started. The modifications have to be defined in 4DScript, the programming language of Enterprise Dynamics.*
Exit	Shuts down Enterprise Dynamics.

Model	Explanation
Create	Shows the library tree and the model layout window. You can build your own model by dragging atoms into the model layout.
Layout window	Shows the model layout window. Atoms can be created in this window either by dragging atoms from the library or by using the Taskbar.
Sub-layout window	*Shows another layout window, but one hierarchical layer lower, e.g. the contents of a composition container.*
Model Tree	Shows the model tree. This 'model tree' gives a hierarchical overview of the atoms displayed in the model.
Library Tree	This option shows all the atoms in the library.

Simulate	Explanation
Run Control	To start, stop and modify the speed of a simulation run.
Clock	Displays the clock.
Run	With shortcut Ctrl-F11 you can run you model
Stop	With shortcut Ctrl-F10 you can stop your model
Stop and Reset	With shortcut Ctrl-F9 you can reset your model
History	Allows you to generate graphs and reports from a single run. The option 'general history' needs to be checked. It is linked to the simulation via the run control window. When graphs and reports of a specific atom are requested, the history of this atom has to be maintained. The easiest way to achieve this is to select the option 'All on'. However, this results in the history collection of all atoms, which can lead to huge data files!
Set stop time	Allows you to define the exact period of a simulation run. This period can be defined in seconds, minutes, hours or days. By selecting the menu option "Reset + Run until Stoptime" in the simulation menu, the simulation will stop exactly at the time entered by the user.
Reset + Run until Stoptime	Resets the simulation and lets it run until the time entered in the menu option "Set stop time" is expired. NB: Next to Run Control, this option is a second way to perform a few simulation runs. However, in the case of an experiment consisting of several runs, the Experiment Wizard is required.
Repetitive	*When this option is switched on, the 100 random generators in Enterprise Dynamics keep repeating the same random figures. Consequently, several exactly similar simulation studies can be performed.*
Antithetic	*When this option is switched on, the 100 random generators in Enterprise Dynamics are set on antithetic. This can help reduce the effect of very extreme values over several runs.*
Seed Value	*Allows you to define the starting value of each of the 100 random generators.*

Results	Explanation
Summary Report	The Summary Report displays an overview of the basic statistics relating to all atoms present in the model, **based on a single run**. NB: For more detailed reports, the Report Atom can be dragged out into the model.
Graphs	Shows various graphs of atoms such as queues, histograms and pie charts, **based on a single run**. These graphs can only be created if the History option relating to the atom in question is switched on.

Experimentation	*Explanation*
Experiment Wizard	Helps you defining your experiment settings and output variables. Afterwards this experiment is run.
Analyze Results	Report definition en Report generation regarding an experiment

Tools	Explanation
Atom Editor	*With the Atom Editor, you can adjust the functionality as well as the appearance of an atom.* *This very effective tool allows you to alter the behavior of existing atoms and to create your own atoms. The programming language 4DScript has to be used here.*
4DScript Interact	*A window in which 4DScript can be entered. Direct execution of the command follows.*
Text Editor	*A simple text editor, the functionality of which can be compared to MS Notepad.*
Cad Import wizard	*Allows the user to automatically generate models from CAD-drawings* *NB: this option has to be purchased separately, so it might be missing.*
GUI Builder…	GUI is short for Graphical User Interface: it allows the user to create his own input fields.
View Atom Labels	This option displays all labels of the selected atom (and of all atoms included in that atom). Labels are variables and attributes the user can allocate to an atom.
Autofit	The Autofit function analyses a data set and searches for the best fit probability distribution.

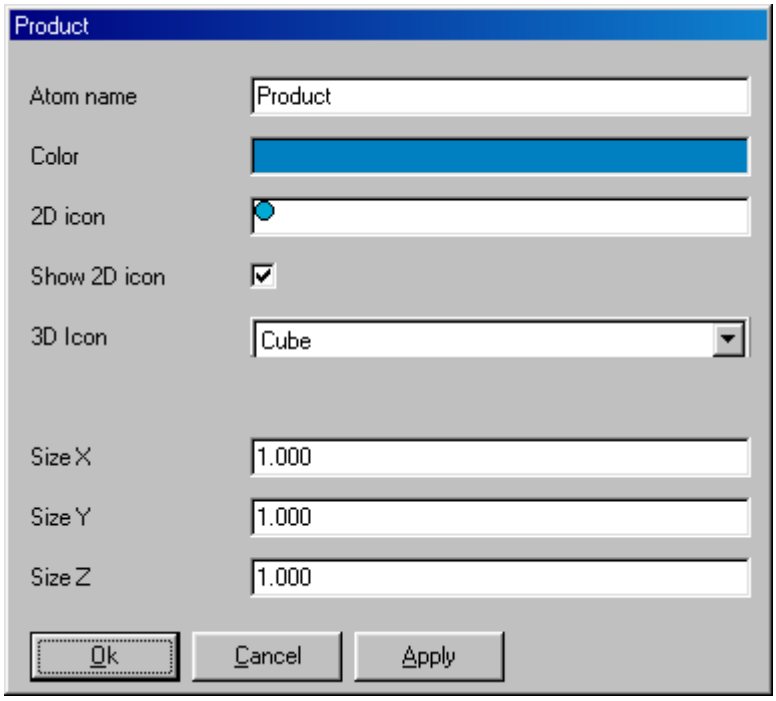
Display	Explanation
2D Model View	Opens the 2D visualization window. *Warning!* In this window, you cannot add atoms to the model or reposition existing ones. For this you must use the Model Layout window.
2D Model Subview	*Opens the same window as the 2D Model View option, but here, only the contents of the selected atom are shown.*
2D Visual Trace	This option displays the movement of products in Enterprise Dynamics. After switching on this option, seven figures appear on the screen, one of which has to be selected. The higher the selected figure, the faster the movements.
3D Model View	Opens the 3D visualization window.
3D Model Subview	*Opens the same window as the 3D Model View option, but here, only the contents of the selected atom are shown.*
3D Background Color	*Opens a color selection window. This option allows you to define the background color for the 3D window. This color will also be used in the VR window.*
3D Shading	*Switches the shadows of 3D objects on and off.*

Window	Explanation
Close all windows	Closes all open windows.
4DScript Overview	Shows an overview of all 4DScript expressions together with an explanation of their syntax. You can also open this window by pressing F2.
Error Monitor	*Opens a window displaying errors encountered in 4DScript.*
Tracer	*Opens the Tracer window. You can enter 4Dscript expressions in this window.*
Layers	*Allows you choose which atoms are visible or not*
Resources Manager	*Opens a window in which the standard atom icons are displayed. You can create and add your own icons!*
Graph Window	*Opens the most recent graph. It is not possible to produce new graphs in this window.*

Help Menu	Explanation
Help Overview	Gives you access to the complete manuals, consisting of the 3 following menu sub-units.
Tutorials	Gives you acces to Tutorials in pdf.format
Add-inns	*Gives you help on the GUI andOptQuest* *NB: OptQuest is an optimization progam and has to be purchased separately, so it might be missing*
About Enterprise Dynamics	Displays information regarding the version in use and about Incontrol Enterprise Dynamics.

Annex 2 Description of a few atoms

1 THE PRODUCT ATOM

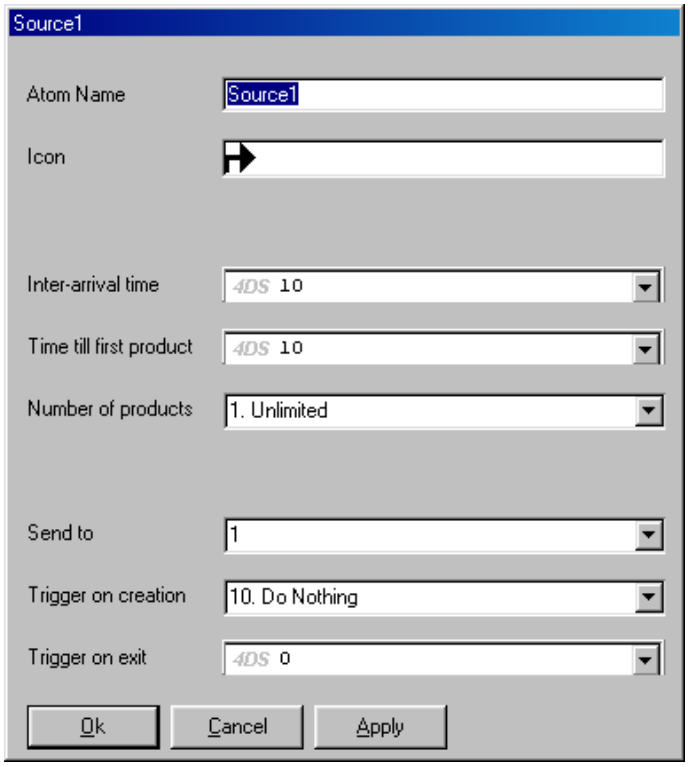


Picture 1-1: The Product Atom

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:

- *Atom name*
  The name given to the atom.
- *Color*
  The color given to the atom.
- *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.
- *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).

2 THE SOURCE ATOM



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:

- *Atom name*
  The name given to the atom.
- *Icon*
  The symbol used to represent the Source atom in the 2D window.
- *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 colour. In Enterprise Dynamics, the colours must be specified either by their number or with the prefix "color" followed by the colour 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.

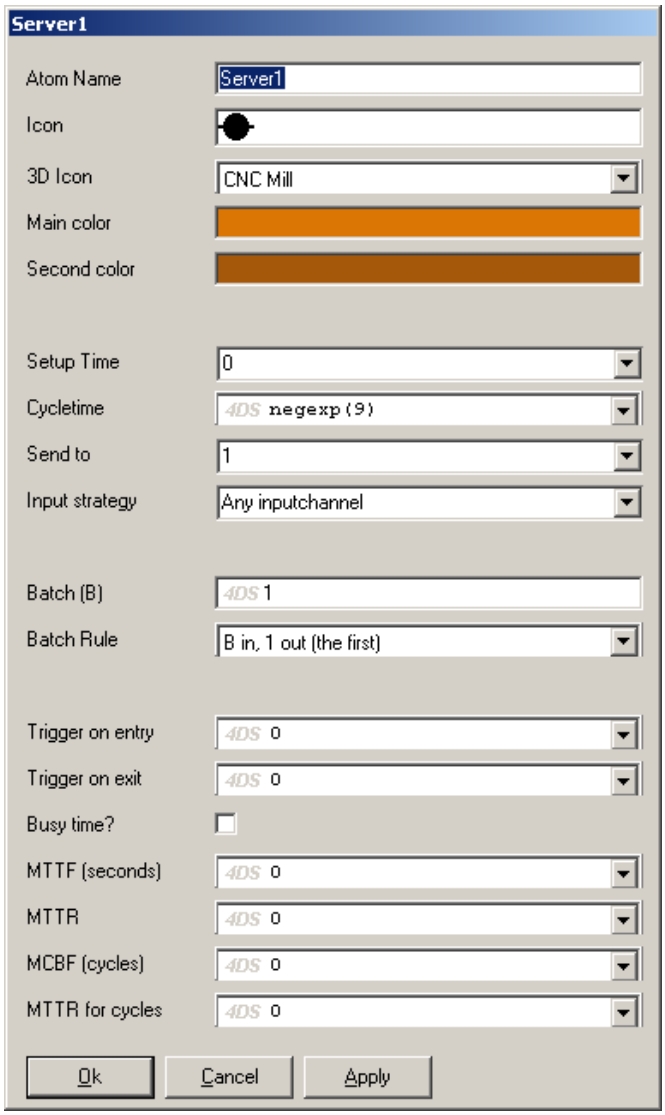
10: *if(=(label([**?**],i),**?**),**?**,**?**).*
A conditional comparison, in which a label's value is considered.

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),set(color(i),colorred),set(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.

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



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:

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

- *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.

- *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.

- *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.

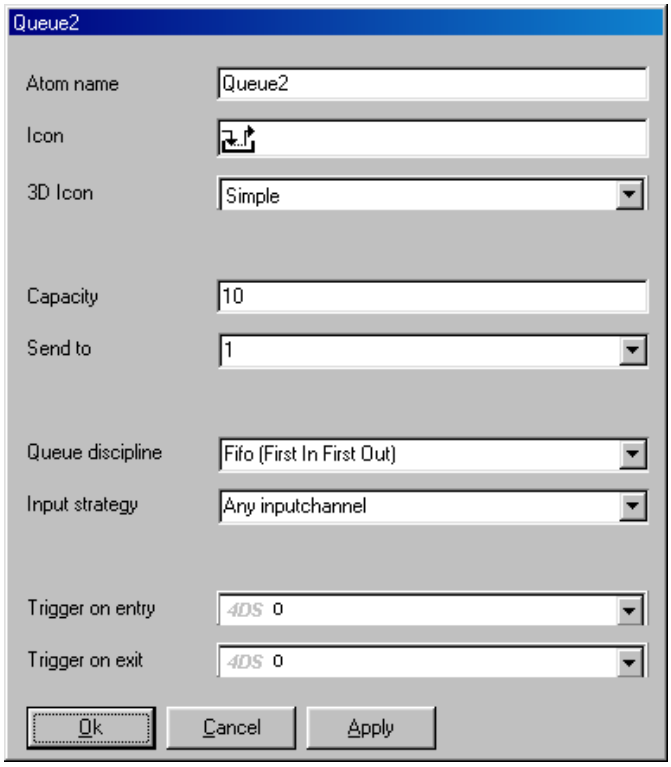
- *MTTF*
  This abbreviation stands for Mean Time To Failure, that is the average time elapsing between the end of a repair and the beginning of next failure. This average time between two Server failures can be defined in the input field. The time must be entered in seconds.

- *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.

4 THE QUEUE ATOM



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:

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

- *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:

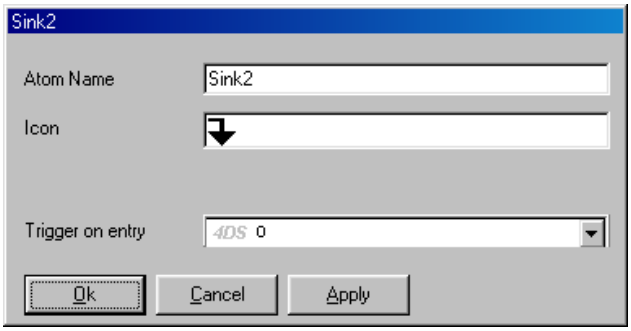
  1: *First in first out.*
  The atoms are put in the queue according to their order of entry.
  2: *Last in first out.*
  The entering atoms are placed at the front of the queue. Consequently, the products leave the queue in reverse of their order of entry.
  3: *Random.*
  This queue discipline places 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.

5 THE SINK ATOM



Picture 5-1: The Sink Atom

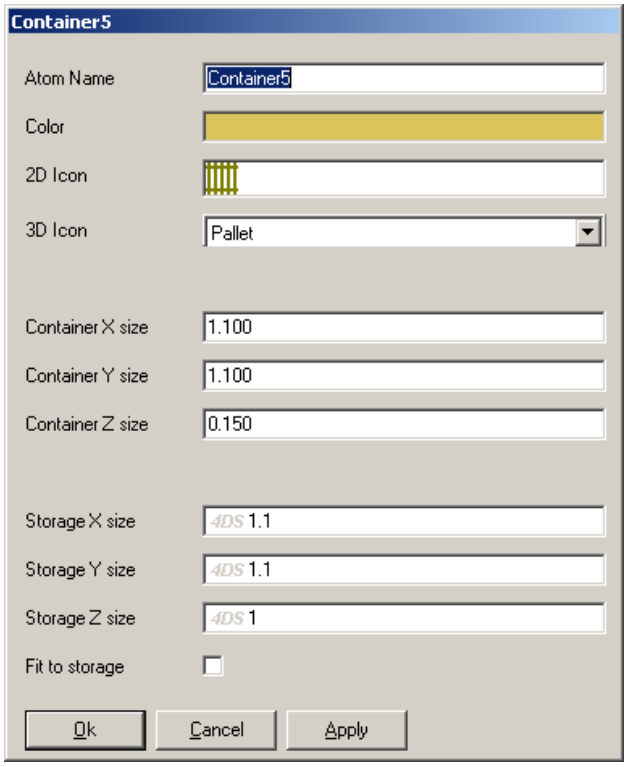
This atom allows products to leave the model. The following settings can be adjusted:

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

- *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.
  For more details: see 'Trigger on Exit' of the Source atom.

6 THE CONTAINER ATOM



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.

- *Atom name*
  The name given to the atom.
- *Colour*
  The colour 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.
- *Use boxvisible*
  Checking this option accelerates the 3D animation, but may result in the 3D icon suddenly disappearing (depending on the size and angle of the 3D animation). The gain in speed is generated mainly from those icons outside the window not being drawn. Particularly in large models it is therefore important to select the "use boxvisible" option.
- *Container X size*

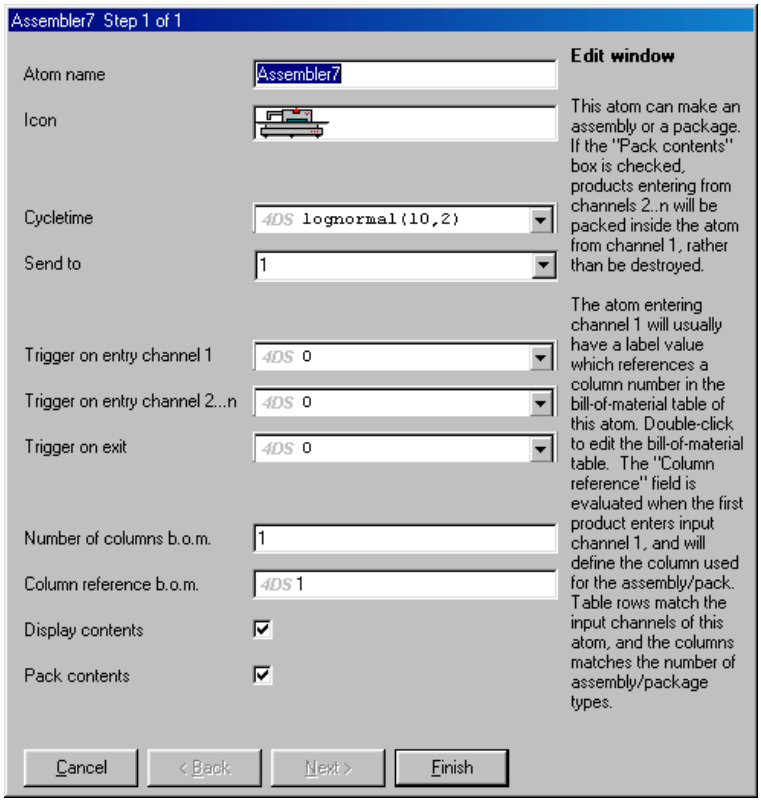
The size of the container measured in the x direction (length).

- *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).
- *Trigger on Entry*
  The command entered in this field is executed when a product is arriving at the Container atom.
  For more information: see Trigger on Exit of the Source atom
- *Trigger on Exit*
  The command entered in this field is executed when a product is leaving the Container atom.
  For more information: see Trigger on Exit of the Source atom.
- *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

7 THE ASSEMBLER ATOM



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.

- *Atom name*
  The name given to the atom.
- *Icon*
  The symbol that represents the Assembler atom in the 2D window.
- *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 not 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.

- *Trigger on entry 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.
- *Trigger on entry 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.

Bill of Material

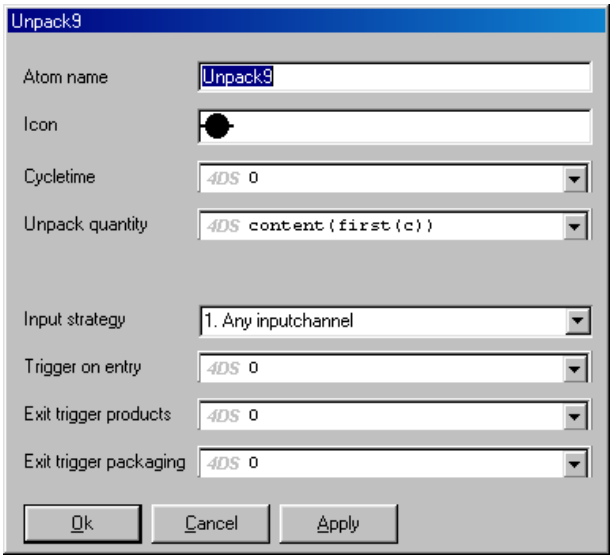
The Assembler atom also has a bill-of-material (b.o.m.) table, (visible by double-clicking the atom). The b.o.m. lists, by input channel, how many atoms are required for assembling the end product.

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.

- *Number of columns b.o.m.*
  Defines the number of columns in the b.o.m.
- *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.
- *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.

8 THE UNPACK ATOM



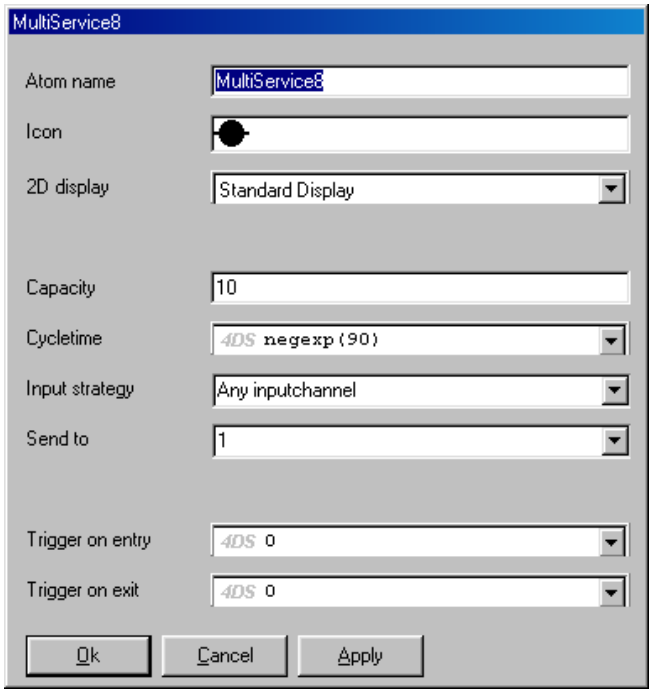
Picture 8-1: The Unpack atom

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.

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

9 THE MULTISERVICE ATOM



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.

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

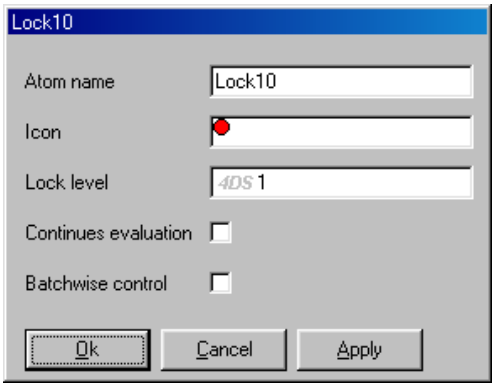
5: *Level horizontal*
Instead of the products, a colored area is visible which scrolls to the right when more products enter the MultiService atom.
6: *Line up content*
The products are visible and displayed one underneath the other.

- *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*

10 THE LOCK ATOM



Picture 10-1: The Lock atom

When the Lock atom alone is used, it functions as a gateway that lets only the number of products indicated in the Lock level through.

When the Lock atom is used in combination with the Unlock atom, it serves to control the maximum number of products in a certain part of the model. The amount of work in process in the model delimitated by Lock and Unlock can then be no more than the so-called Lock level.

Due to the connection between the two, the Unlock atom is described below.

- *Atom name*
  The name given to the atom.
- *Icon*
  The symbol used to represent the Lock atom in the 2D window.

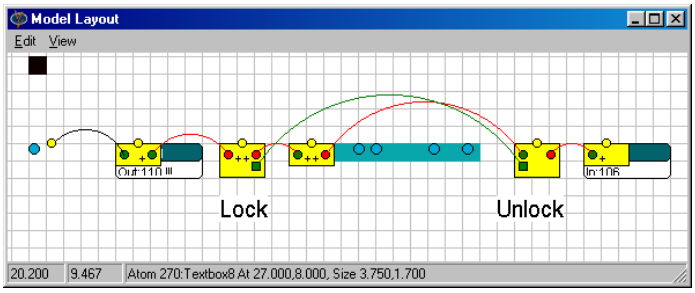
- *Lock level*
  When the number of atoms between Lock and Unlock reaches the specific level entered in this field, the input of the Lock closes, thereby blocking access. When Unlock is not added, the input closes without reopening.
- *Continues evaluation*
  When this option is checked, the field Lock level is evaluated every time when a product enters the Lock atom, instead of being evaluated only after a model reset.
  Only when the field Lock level contains a 4DScript command (and not only a number) may it be necessary to activate this option.
- *Batchwise control*
  When the Lock level is reached, access via the Lock is blocked and reopened only after all products in the channel have disappeared via the Unlock atom. This creates a form of batch processing with the Lock level as batch size.

11 THE UNLOCK ATOM

This atom has no settings and is used only in combination with the Lock atom. Each time when a product leaves the Unlock atom, the Lock atom allows a new product to enter the system (unless the “Batchwise control” option is activated); in that case all products must first have passed through the Unlock atom).

In the example below, the Lock and Unlock atom control the quantity of products on the conveyor. See how both atoms are placed in the product route `in a normal way' and how they are connected via their second output or input channel respectively.

The Unlock atom has no window that needs filling in. A Lock atom can be connected with several Unlock atoms.



Annex 3 A first start in 4DScript

4DScript is the programming language of Enterprise Dynamics. Everything that is executed in Enterprise Dynamics is or can be done via 4DScript.

This document provides for beginners the structure of 4DScript and a list of often used commands, illustrated with examples.

We start with the syntax rules together with the mathematical and logical operators. Second, the concept of referencing is illustrated with examples. The third part consists of the most common commands in ED.

1. The basics of 4DScript

The basic syntax of 4DScript is simple and is valid at any position where you can write 4DScript:

1 the language contains a number of words (commands);
2 these 4DScript commands can have (up to 255) parameters;
3 the parameters are placed between parenthesis ( );
4 parameters can be *values*, *strings* or *expressions* (other 4DScript words)
5 parameters will always be separated by commas;
6 if a parameter should be interpreted as a string, the string parameters are always placed between square brackets [ ]. If the parameter should be executed as 4Dscript code, the parameter is written in a normal fashion;
7 comments can be placed between squiggly brackets { }.

These rules apply everywhere and are always valid. However, for some statements these rules do not result in very readable code. In order to make parts of the code easier to understand for others, the 4DScript syntax makes an exception in some cases. This applies in particular to mathematical and logical symbols.

Mathematical symbols

Consider the following valid statements:
+(12,7) results in 19
+(/(100,10),6) results in 16

A more natural way:
12+7
100/10+6 or 6+100/10

All the special mathematical operators are evaluated in the following order:

*	=	multiplication
/	=	division
+	=	addition
-	=	subtraction

The normal priority rules for these operators apply. In case you have doubts, use parenthesis ()!

Logical symbols

For logical operators (like >, < , =) it is also possible to disregard the rule that the 4DScript command needs to be written first and then the parameters

Consider the following valid statement:

>(10,6) can also be written in the `normal' fashion as: 10>6
<=(23,12) can also be written as: 23<=12

This example can be extended to more 4DScript logical symbols:

=	=	equal
>	=	larger than
<	=	smaller than
>=	=	larger than or equal to
<=	=	smaller than or equal to
<>	=	not equal

and
syntax: and(e1,e2)

Returns 1 if e1 and e2 are true (1), returns 0 if not. If e1 is not true, then e2 is not evaluated. Instead of writing and(e1,e2) the user can also write e1 **and** e2.
See also **or**

So, both commands are used as usual…

max
Syntax: max(e1,e2)

Returns the maximum of e1 and e2. See also **min**

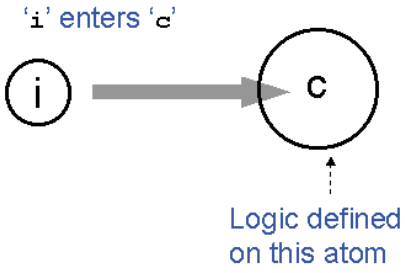
We end this chapter by a few commands on time and time conversion. Remember that ED `thinks' in seconds, but it is easy to convert time:

time returns the current model time in seconds
mins(e1) returns e1 (minutes) in seconds: multiplies e1 with 60.
hr(e1) returns e1 (hours) in seconds: multiplies e1 with 3600.

2. Referencing

A very important subject in ED is referencing: if we are talking about the `next' atom, what is our viewpoint? It gets more complicated if there are atoms contained in other atoms like products in a queue.

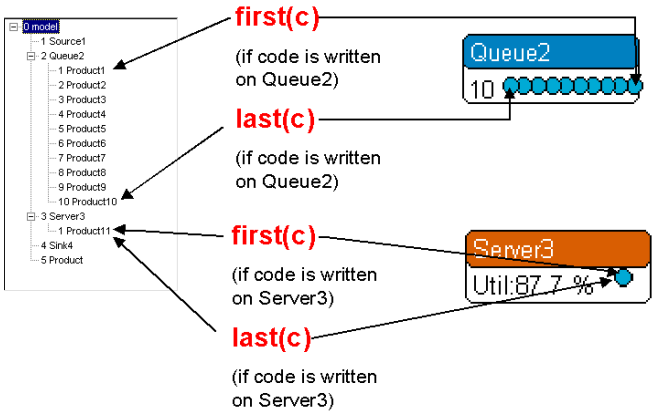
We have seen the letters c and i in different statements, now we will explain them:
`c' refers to the **c**urrent atom where the statement is written on
`i' refers to the **i**nvolved atom, the one entering or leaving the current atom



Picture 1

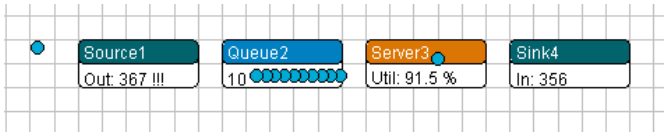
To keep it simple: the `i' is used only in statements written on Trigger on Entry or Trigger on Exit!

The key to understanding the concept of referencing is the next example:



Picture 2

On the left of picture 2 you see the model tree of a simple queuing system, frozen at one moment in time:



Picture 3

On the same level we find the Source, the Queue, the Server and the Sink and the Product atom placed before the Source (number 5 in the model tree, because it is mostly added after you reset your model the first after making the lay-out).

Queue2 contains 10 products, named Product1 to Product 10. They form a second level, compared to the Product, Source, Queue, Server and Sink. Of course Product1 and, lets say, Product5 are on the same level. Server3 contains 1 product, named Product11.

The highest level is the model itself. In picture 1 you can see this in the model tree! Can you predict how the model tree changes if you reset this model?

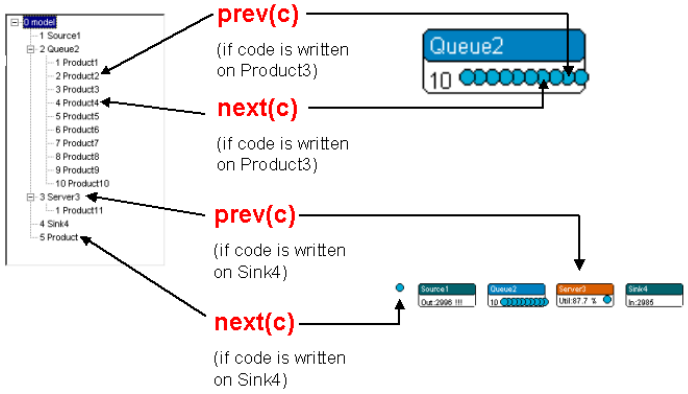
If you understand the hierarchy in this model, you will understand referencing…

First a few commands which are often used for referencing:

first(e1) refers to the first atom inside atom e1
last(e1) refers to the last atom inside atom e1
next(e1) refers to the next atom on the same level as atom e1
prev(e1) refers to the atom in front on the same level atom e1

So first(c) refers to the first atom inside the current. If the current is a Queue it could be the first product in the queue. Now look at picture 1 again and the examples on *first* and *last.*
So, first and last are always looked at one level deeper from the atom where the statement is written on!

The commands next and previous are explained below:



Picture 4

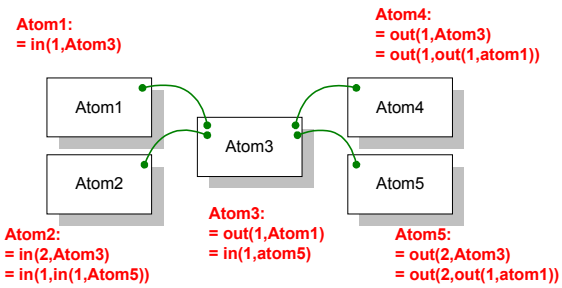
Now look at picture 3 and the examples on *prev* and *next.*

So, prev and next are always looked at the same level from the atom where the statement is written on!

There are a few commands often used for flow control and related to channels:
out(e1,e2) refers to the atom connected to output channel e1 of atom e2
in(e1,e2) refers to the atom connected to input channel e1 of atom e2
openoutput(e1) opens the general output of atom e1
closeoutput(e1) closes the general output of atom e1. If closed, the channels cannot be ready, regardless of individual channel settings.

In a similar way you can define **openinput** and **closeinput.**

Take a look at picture 5 regarding the use of *in* and *out:*



Picture 5

Example 1:
So refers *in(1,c)* (written on Atom3, the point of view) to Atom1, but the same statement written on Atom4 refers to Atom3!

Example 2:
content(in(3,c)) returns the content (number of atoms) contained in the atom which is connected to input channel 3 of the current atom.

Example 3:
closeoutput(in(2,c)) closes the output of the atom connected to the second input channel of the current

Notice the way commands are nested…

3. Important commands

In every programming language there are important commands, which perform as some kind of building blocks.

3.1 For conditional statements: **if**

Syntax: if(e1,e2 {,e3})
Executes e2 if e1 is true (1); otherwise executes e3 (if specified). Returns result of e2 or e3.

Example 1:
if(time>3600, msg[more than an hour], msg[less than an hour])

Translated: if time>3600 seconds then give message `more than an hour' else give message `less than an hour'.

Example 2:
if(content(c)>10, closeinput(in(1,c))

Translated: if the content of the current atom is more than 10 then stop the input of the atom connected with the first incoming channel else do nothing

Don't worry about the used commands like msg, content or closeinput. We'll explain them later!

3.2 For executing more than one statement: **do**

Syntax: do(e1,e2,...,e25)
Executes e1, e2, etc. in order of sequence. Returns result of last expression. You can use up to 25 parameters. If more parameters are needed, several do loops can be nested.

Example:
do(set(color(i),colorred),
set(icon(i),2),
setlabel([temperature],uniform[20,40],i)

Three things are done at the involved atom (mostly a product): the color is set to red, the icon is changed to iconnumber two and a label with the name *temperature* is stamped on the product, its value is chosen randomly between 20 and 40.

3.3 Labels

The use of labels is the way to store local (temporary) variables by users. They are mostly attached to products and can represent a weight, a customernumber, a productiontime etcetera. They are defined with the command *setlabel* and reproduced with *label*

a. setlabel(e1,e2,e3)

Defines a label on atom e3 where e1 is the name and e2 is the new content of the label. Labels do not have to be created, at the moment they are referenced they exist. The *sddb* command does the same.

Example 1:
setlabel([Weight],10.24,i)

First select an atom and create (and give a value to) a label with the command setlabel. Suppose the name of the label is v1 and the value is 100:

Example 2:
setlabel([v1],100,animatom)

label([v1],animatom) now returns the specified value (100) of the selected atom.

b. label(e1,e2,{e3})

Returns the contents of label named e1 defined on atom e2. Labels do not have to be created, at the moment they are referenced they exist. The label name e1 is always a string and is case sensitive.

The result is a string or a value, depending on the contents and on e3: if e3 is not specified or e3=0 then the result is a value if the contents can be converted to a value, otherwise the result is a string. If e3=1, the result is always a value. If the contents cannot be converted to a value the value is 0. If e3=2, the result is always a string. If you use long names and many labels you lose speed.

To avoid mistakes: use only lower-case letters, because labels are CASE-SENSITVE!

3.4 We conclude with a few other important commands:

age
Syntax: age(e1)

Returns the age of atom e1. The age is the difference between current time and creation time. When a run is reset, the creation time of all atoms becomes 0.
See also **maxage, minage, avgage**

content
Syntax: content(e1)

Returns the contents (the number of atoms contained at the highest level) of atom e1.
See also **maxcontent, avgcontent**

avgstay
Syntax: avgstay(e1)

Returns the average staying time of atoms in atom e1 over the total runtime.

input
Syntax: input(e1)

Returns the input of atom e1. The input is defined as the number of atoms that have entered atom e1 until now. The opposite is the output of the atom.
See also **output**