


[Tutorials](#) > [Supply Chain GIS \(AB + DE\)](#)

Phase 5. Processing orders at the distributor

Let us continue defining the order processing in this phase, now on the distributor side.


We will build a flowchart using the Process Modeling Library that manages the orders. Also, in this phase, we will finally create trucks (as you remember, the population on  Main is still empty) and make them take part in the process as resources.

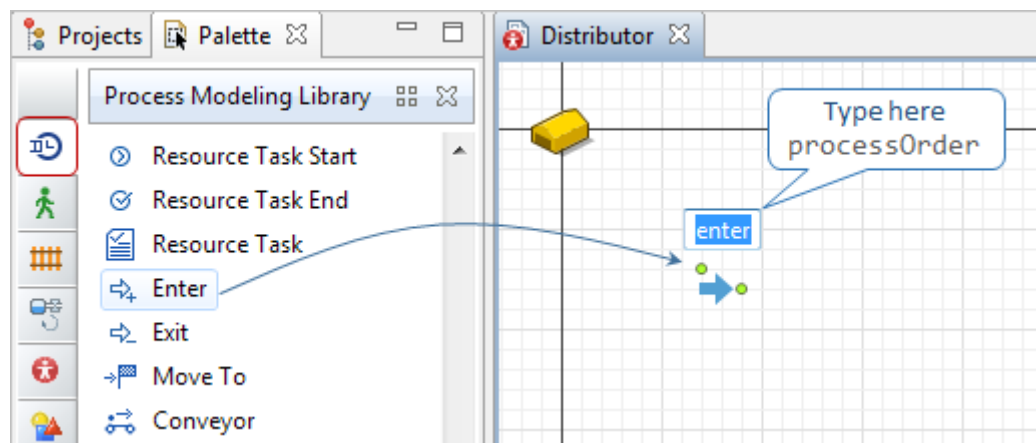
In the previous phase, we have set up an event that sends messages, now we will ensure the message delivery.


To insert orders into the process flowchart

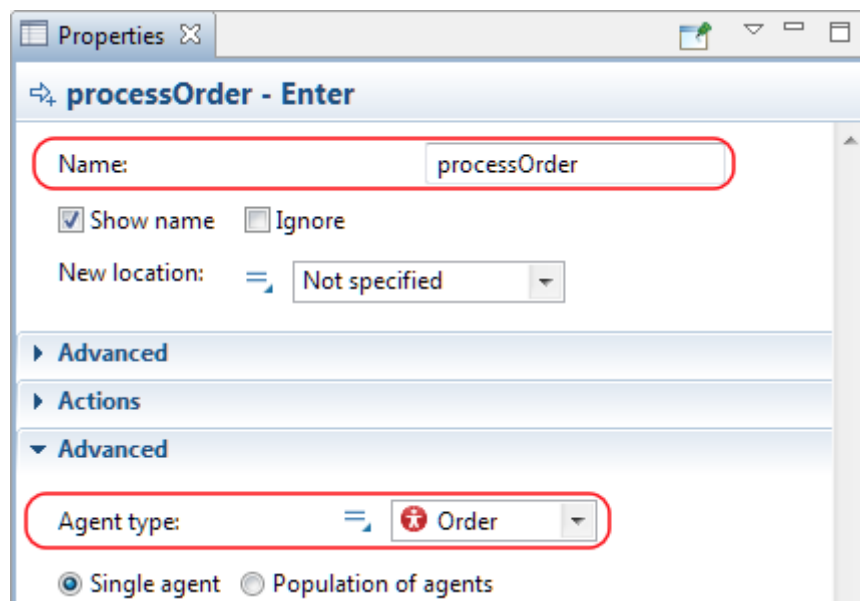
1. Go to the **Projects** view and double-click the  **Distributor** agent type to open its diagram.
2. Switch to the **Palette** view and open the  **Process Modeling Library**.


The [Process Modeling Library](#) contains objects that you use to build flowcharts that depict process and define the parameters and actions of agents and resources in the model.

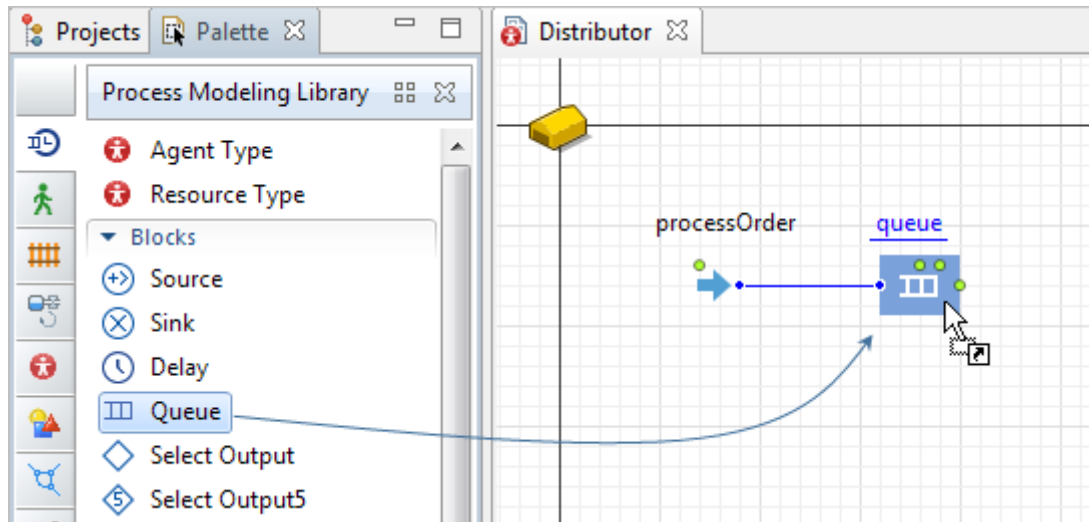
3. Start building the process flowchart with the library object  **Enter**: drop it onto the diagram from the palette. This object inserts already existing agents in the process flowchart, and it is a common way to implement [the process-centric modeling method in the GIS environment](#).



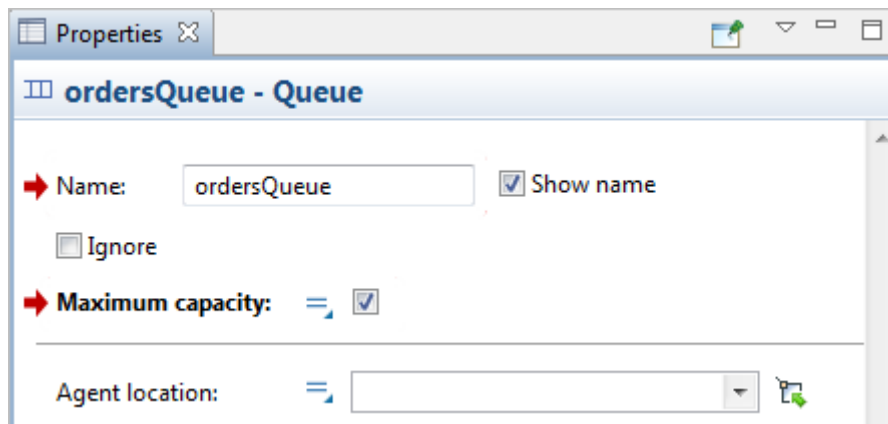
4. Name the object `processOrder`. We need to specify what kind of agent we want to insert in this flowchart:  **Order**.



5. Next, drag the  **Queue** object and let it connect with `processOrder` as shown in the figure below. When you place one library object close enough to another, they automatically connect, and you only need to follow the direction of the flowchart: from left to right.




6. The orders will be first stored in the queue that allows **maximum capacity**, we do not need any limits here in our model:

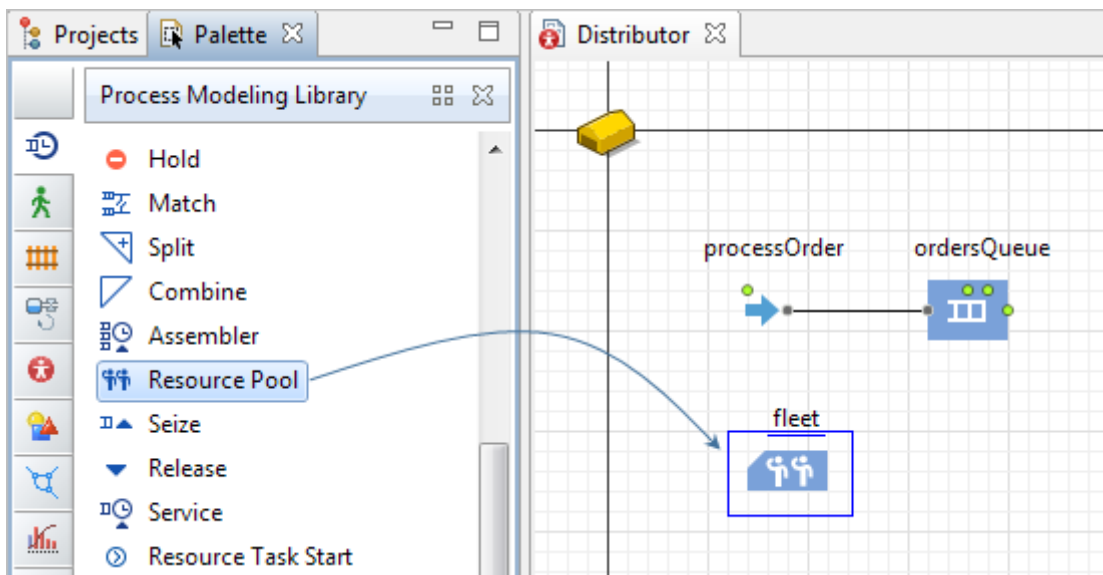


7. We will now continue building the flowchart. The second part will manage the resources: trucks in our model.

To define truck management


1. Let us now add the  **ResourcePool** object that defines the resources' properties. This library object is a bit different from other objects because we do not need to connect it to the flowchart with connectors. We will select it in another object's property.

Name the object `fleet` and go to its **Properties** view.



2. And now we will use this library object to create and define the trucks in our model. By default, the **Resource type** is set to *Moving* which suits us perfectly.

Next, we can define the number of the trucks in our model by *directly* specifying the **Capacity** parameter. For this model, 5 trucks should be enough for our supply chain.

None of the above will matter, if we do not tell the **Resource Pool** what kind of resources it should create and manage. Specify our agent  **Truck** as the **New resource unit**, and then we will be able to use the trucks as resources in the flowchart.

Here we can also specify the **Speed** parameter and assign a realistic speed value to the trucks in the desired units; for instance, *70 kilometers per hour*.

3. The last parameter of the trucks as resource units that we need to specify is their correct population. We want this **Resource Pool** to populate the `trucks[...]` agent. Expand the **Advanced** section of the `fleet` properties and choose to **Add units to: custom population**.

Then, specify the original population on  **Main**:

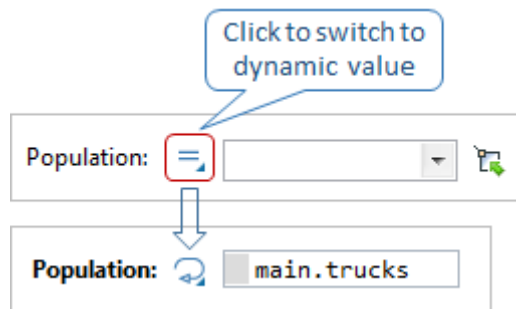
Advanced

Add units to: ☐ default population ☒ custom population

Population:

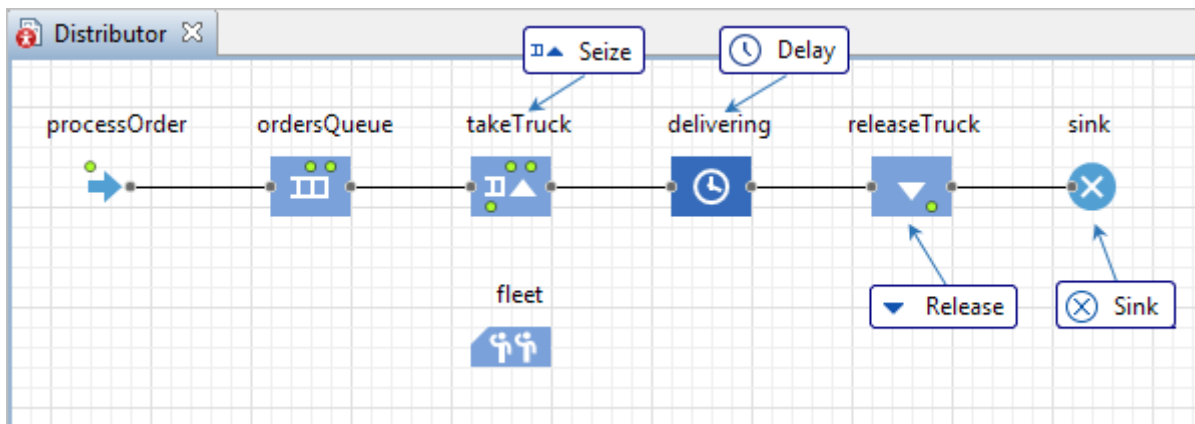
Force statistics collection: ☐

- When AnyLogic suggests that you select some element from the drop-down list, but you need to type a piece of code in the parameter, click its icon to switch to the parameter's dynamic value mode:



- Let us continue building the process flowchart from where we have stopped, `ordersQueue`.

Add these four library objects, **Seize** – **Delay** – **Release** – **Sink**, and name them as shown in the figure:



- The **Seize** object seizes the resources from the given **Resource Pool** object to utilize them. Select our `fleet` in the **Resource sets** property, then this object will use the resource units from this resource pool (our trucks).

Enable the **Maximum queue capacity** - the queue where orders wait for an available truck.

Properties

takeTruck - Seize

Name: ☒ Show name ☐ Ignore

Seize: ☒ (alternative) resource sets
☐ units of the same pool

➔ Resource sets:

☒ ☐ ☐ ☐ ☒

☒ Add list

Seize policy: ☒ Seize whole set at once
☐ Seize units one by one

➔ Maximum queue capacity: ☒

6. Expand the **Actions** section, and enter the following code in the action **On seize unit**: `send (agent, unit);`

When this object seizes a truck (`unit`), it sends an order (`agent`) to this truck, and the order, as you know, contains the information about the retailer that has sent it.

takeTruck - Seize

Resource sets:

☒ ☐ ☐ ☐ ☒

▼ Actions

On enter:

➔ On seize unit:

On prepare unit:

▼ Advanced

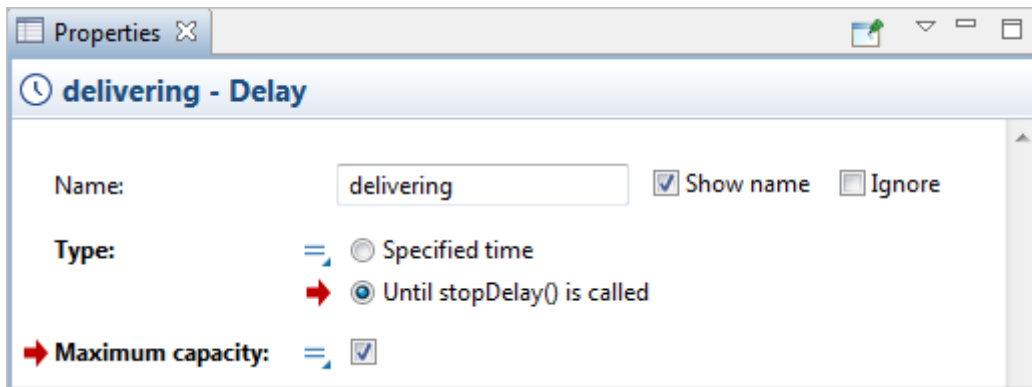
Agent type:

When you click in the edit box of any code parameter, you can see a small light bulb icon in its upper left corner. Hover over it to see the list of possible local variables.

Local variables here, `agent` and `unit`, allow you to access the **Agent type** and the **Resource type** correspondingly that we manage with this flowchart. We have selected the agent type (`Order`) in the `processOrder`, and it has propagated through the following flowchart blocks as the agent type whose process they define. The resource type (`Truck`) was set up in the **Resource Pool** that this **Seize** object uses.

7. The **Delay** object, that we called `delivering`, presents the time interval that it takes for a truck to deliver the product. Since our retailers are located in different places, we do not know exactly how long it is going to take to get to them. That is why the delay ends when the truck arrives and signals for it. We will model this action in the next phase.

Now, we set the **Type** of delay to *Until stopDelay() is called*, and enable **Maximum capacity** to store orders without restrictions. That means, we have to call `stopDelay()` somewhere else.

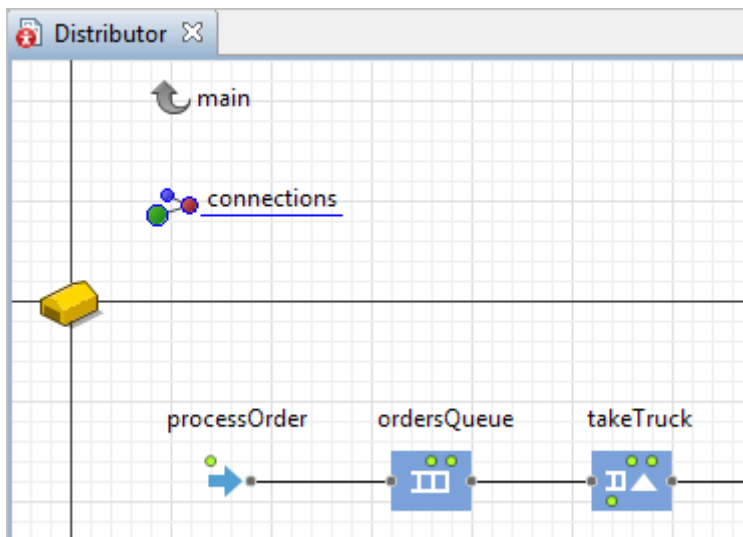


8. The **Release** object naturally releases the resources that were taken by **Seize** when those units are no longer needed. Before the agents are disposed at the end of the flowchart, they have to release all of their resources.

The **Sink** object disposes of agents and ends a flowchart.

To enable correct message delivery

1. Move the diagram down to find the connections object. This object is present on every agent's diagram; it manages the links to other agents in the model and supports [communication between agents](#).



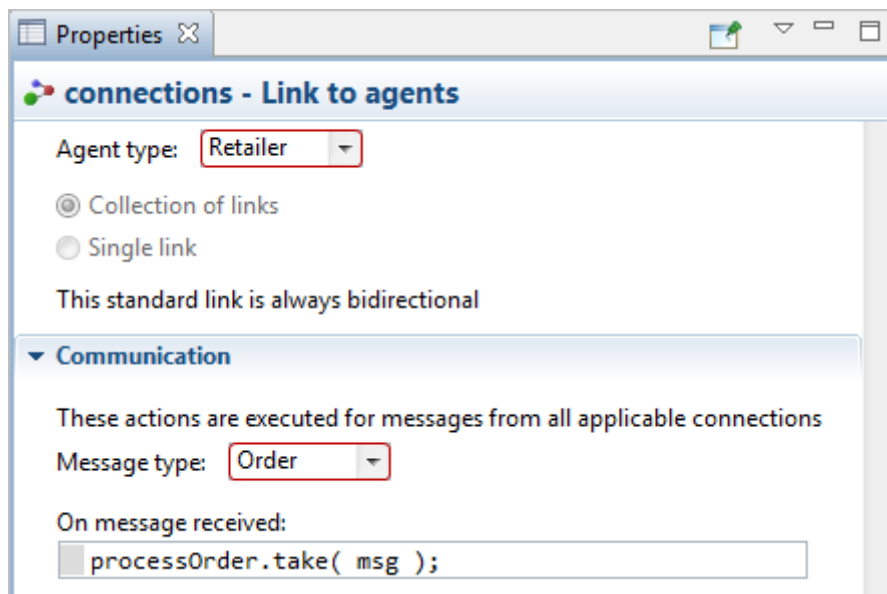
2. Communication between agents happens when they send and receive messages from each other. In the previous phase, we have created the event with the action that sends messages to the distributor. Now we will make sure the **Distributor** agent receives the messages and processes them correctly.

The messages are sent from the **Retailer**, and that is why we need to choose it in the **Agent type** property. What we are sending is of the **Order** agent type, and we choose it as the **Message type** here.

We have set up the message delivery of the special type we need. Next, let us tell **Distributor** what to do when it receives a message. Type the code line `processOrder.take(msg);` in the property **On message received**.

`take(agent)` is a method that belongs to the **Enter** library object (follow the link to read about its functions), which we named `processOrder`.

When the message is received, the agent of type **Order** is inserted into the `processOrder` block, and the flowchart starts processing it, i.e the distributor sends a truck to the retailer who demanded the product delivery.



In the next phase, we will finally define the trucks' movement logic, from the distributor to one of the retailers and back.

Reference model: [Supply Chain GIS - Phase 5](#)

◀◀ [Phase 4. Sending orders from the retailer](#)

▶▶ [Phase 6. Defining trucks' movement](#)