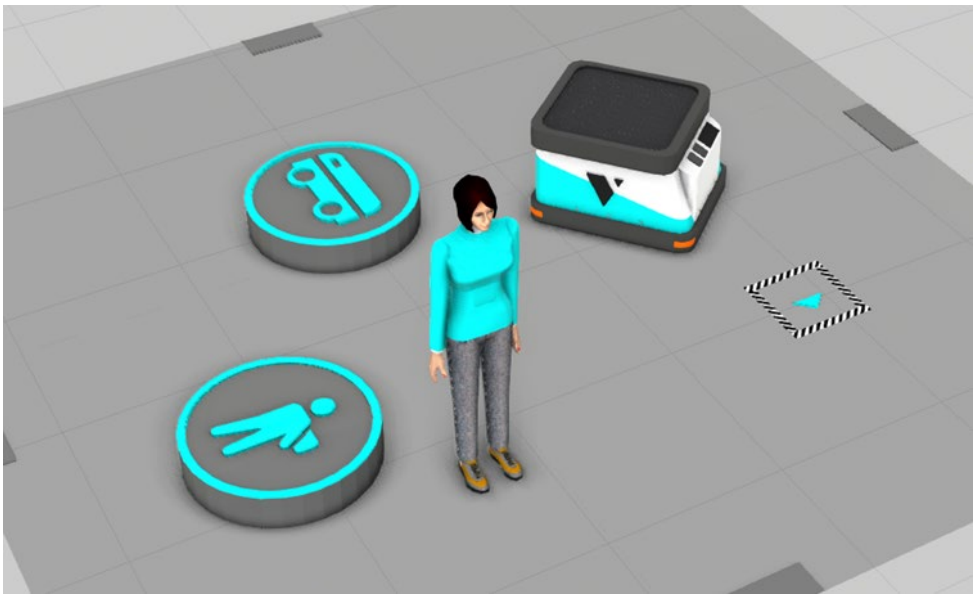
Process Modeling Resource - Manual

Visual Components 4.8 | Version: November 30, 2023



This tutorial is an introduction to the features of Process Modelling Resources and Transport Controllers. It requires a basic knowledge of Process Modelling with Visual Components.

This tutorial contains:

- Features overview
- Use case examples
- Resource properties

Contents

General Features

Resource priority

Available resources are prioritized for transport and work tasks according to the "ResourcePriority" property in the Transport Controller.

Priorities

Nearest: Available resource nearest to the collecting (pick) or work location is allocated for the task

Least Utilized: Available resource that is least utilized (Statistics.Utilization) is allocated for the task

Shortest Travel: Available resource that has the shortest travel distance along the pathways to the collecting (pick) or work location is allocated for the task.
Note: Using this option may negatively affect the simulation performance.

None: Arbitrary priority

Transport and Work priority

A "Priority" property in a transport link, determines transportation priority via the corresponding link among all active transportation tasks. Similarly, all work tasks are prioritized according to the "Work::Priority" property in the Transport Controller. Transport and work tasks share the same priority system.

Link priority is applied to collecting. See section Generic resource properties and under "Transport" for reversing delivery order.

The priority is a number. The lower the value, the higher the priority, e.g., 1 = high priority and 100 = low priority. Tasks with equal priorities are dealt with the First-In-First-Out (FIFO) principle.

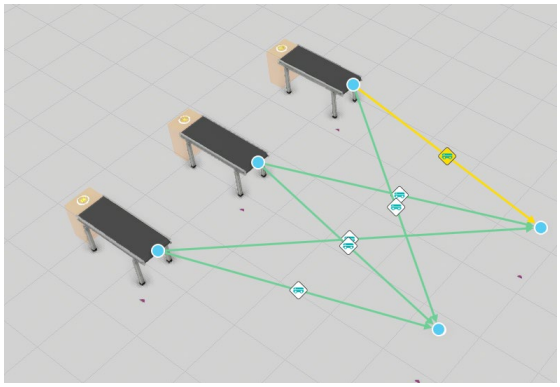
Note: The priority is applied only to the active transport/work tasks. This means that it does not affect the transportation pairing between the processes. In other words, processes do not prioritize the transportation/work that the processes send/publish to the transport system.

Multi-transporting

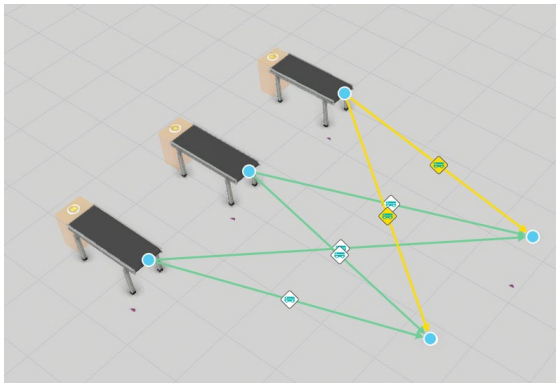
Resources can collect and deliver multiple products simultaneously between processes that have active transports (matching TransportIn and TransportOut statements).

Resources have a property "Transport::Capacity" which defines the maximum amount of products that can be on-board at a time. Resources collect all products first, then deliver. A strategy defined by a "Strategy" property in the Transport Controller is followed when multi-transporting. The selected strategy is applied to all resources connected to the Transport Controller.

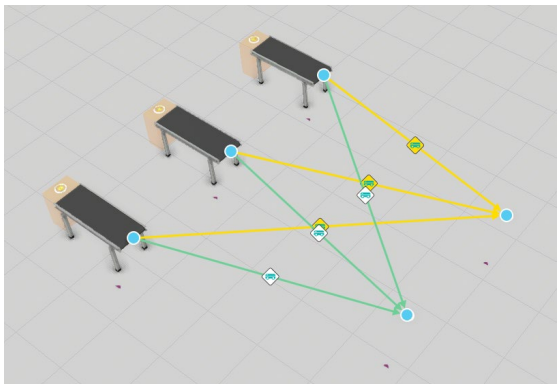
Strategies



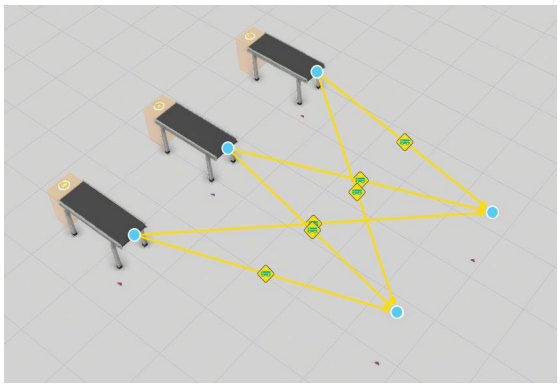
One-to-One: Collecting and delivering is carried only between two processes at a time



One-To-Many: Collect all from one process and deliver to any number of processes at a time



Many-To-One: Collect from any number of processes and deliver to one



Many-To-Many: Collect from any number of processes and deliver to any number of processes

Collecting and delivering

A resource can receive new collect tasks until its capacity is consumed or it initiates delivery. After delivery is initiated, new collect tasks are not accepted until all deliveries of the collected products have been completed.

Delivering the collected products is carried in the order defined by a property "Transport::LIFO" in the resource. If LIFO (Last-In-First-Out) is enabled, the products are delivered in reverse order to which they were collected.

Note: Multi-transporting is *greedy*, which means that a resource will be dispatched to handle as many transportation (collect and delivery) tasks as possible when multiple transportations that meet the strategy are available.

Tools

In each transport link, a tool used to transport the product(s) can be defined. Work tasks have a Tool definition in the transport controller "Work::Tool" property. The options are:

Use Current: Keep existing tool (if any used in the previous task)

Tool Name: Define the tool by its (component) name. The name does not need to be complete. A part of the name is sufficient and is used as a lookup name. In search, (*) wildcards are added before and after the name. For example, if two tools are available with names *MyTool #1* and *MyTool #2,* the tool name *MyTool* would accept either one.

Product Property (not available in Work)**:** The Tool Name can be read from the given product property (string). The name lookup is the same as in the **Tool Name** option above**.**

No Tool: No tool is allowed. If the resource has any existing tool, that must be returned first.

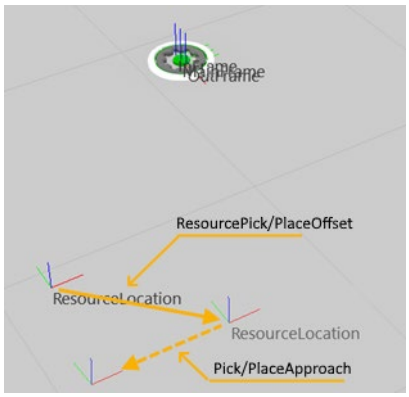
Most Transport Controllers require the available tools to be connected through the *Tools* interface. An error is printed to the Output, and simulation is paused, if the tool is not found.

If the defined tool is found but reserved for another resource, the task is kept in a priority queue until the tool is available. If other tasks are available that do not require the tool, the resource will be dispatched to that task instead (skipping tasks with a higher priority).

Approach and resource position

A ResourcePosition is defined in each TransportIn, TransportOut, and Work statement. The ResourcePosition is the final target where the resource stops and picks/places the products from/to the process.

The ResourcePosition's location can be translated with the "ResourcePickOffset" and "ResourcePlaceOffset" properties in a transport link by enabling "UseCustomParameters" or by defining the default values in the Transport Controller "LinkDefaults" property tab. The defaults are applied to all links that do not use custom settings.



A via point, known as approach position, can be defined similarly to ResourcePosition in each link and the defaults tab of the Transport Controller. The approach position is applied as a translation relative to the ResourcePosition (final target).

If the approach is defined, it will also be used as a via-point when leaving the process.

Idling and charging

Idle location, idle path and charging stations are defined by placing corresponding components around the layout and connecting them to the Transport Controller through the "Idle/ChargingLocations" interface.

Idling stationary

Resources proceed to the nearest free idle location after being on standby (without tasks) for a duration defined in each resource's "TimeToIdle" property.
Note: If the time is set to zero, idling is disabled.

Resources can divert to a new destination if they receive new tasks while traveling to an idle location.

Only one resource can occupy an idle or charging location at a time. If all locations are occupied, or no locations are found, the resource will idle in-place and print the following warning to the Output panel:

[Human (Anna) - WARNING]: No idle locations available. Idling at place.

However, the resource will be notified as soon as any of the locations are released, and it will go automatically to continue idling at that location once it has successfully reserved it.

Idle locations can be used for charging if the "Charger" property is enabled.

Idling on path

Resources can also follow a certain path in a loop when being idle. In that case, the Idle Path component must be connected to the Transport Controller through the "Idle/ChargingLocations" interface. Then two or more waypoints must be added and connected either manually, or by pressing the "Add Waypoint" button in the component properties. The "OrderIndex" property value of the waypoints defines the direction of the path starting from the smallest index.

Note: The resources cannot be charged while idling on path.

Charging

Charging locations are used to increase the "Power::CurrentCapacity" property's value for the resources that have the property "Power::Enabled" selected. The resource's property "Power::LowBattery" is set to True and the resource will seek an available charging location after its "Power:: CurrentCapacity" property's value falls below the set threshold defined by "Power::ToChargeLimit", which is a percentage of the maximum capacity.

The resource does not immediately proceed to charge if it has active transport tasks. In that case, it will finish the tasks and then proceed to charge. If the resource is reserved for a process, it can only proceed to charge once released.

At charging, the resource will be occupied until its available capacity has reached the percentage value defined by “Power::ToChargeLimit” of the maximum charge defined by "Power::Capacity". If no new tasks are available after the resource has been charged above the limit, the resource will stay and continue charging until it has reached maximum charge.

When at maximum and "AllowIdling" is not enabled, the resource leaves the charging location and proceeds to the nearest idle location to wait for new tasks.

If the property “Power::ChargeOnIdle” is enabled, the resource will seek an available charging station instead of idle location when being on idle. Once a charging station has been reached, the resource will be charged (its state is “Charging”). However, the resource **is not forced to stay at the charging location** until the “Power::ToChargeLimit” has been reached, but can leave at any time a new task has been received.

A charging station can be used also as an idle location, if the "AllowIdling" property is enabled. In that case, if a resource happens to arrive to a charging station for idling, the resource is being charged (its state is “Charging”) when it stays at the station. However, the resource **is not forced to stay at the charging location** until the “Power::ToChargeLimit” has been reached, but can leave at any time a new task has been received.

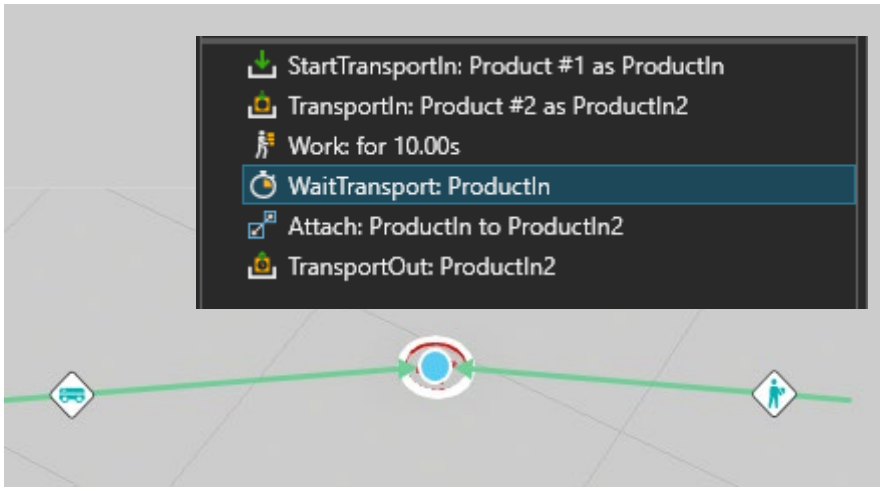
The first available charging station is reserved according to the order defined by the “Power::ChargerPriority” property. When set to Nearest, the resource will seek the closest charging station. When set to “Local Priority Order”, you can define the priority order of charging locations in the resource’s “Power::ChargerPriorityOrder”. When set to “Global Priority Order”, you can define the priority order with the Transport Controller’s property “ChargerPriorityOrder”.

If the battery runs out when all the charging locations are occupied, the “Power::OnLowBattery” property defines if the resource should stand still or go and wait at an idle location. However, the resource will be notified as soon as one of the stations will be released, and it will go automatically to charge at the station once it has successfully reserved it.

WaitForTransport

A "WaitForTransport" property in a transport link can be used to hold the product on-board/in hand until the receiving process starts to execute a Wait Transport statement. If the Wait Transport statement is already being executed once the resource arrives, the resource will immediately place the product into the process. Otherwise, the resource will wait at the designated ResourcePosition.

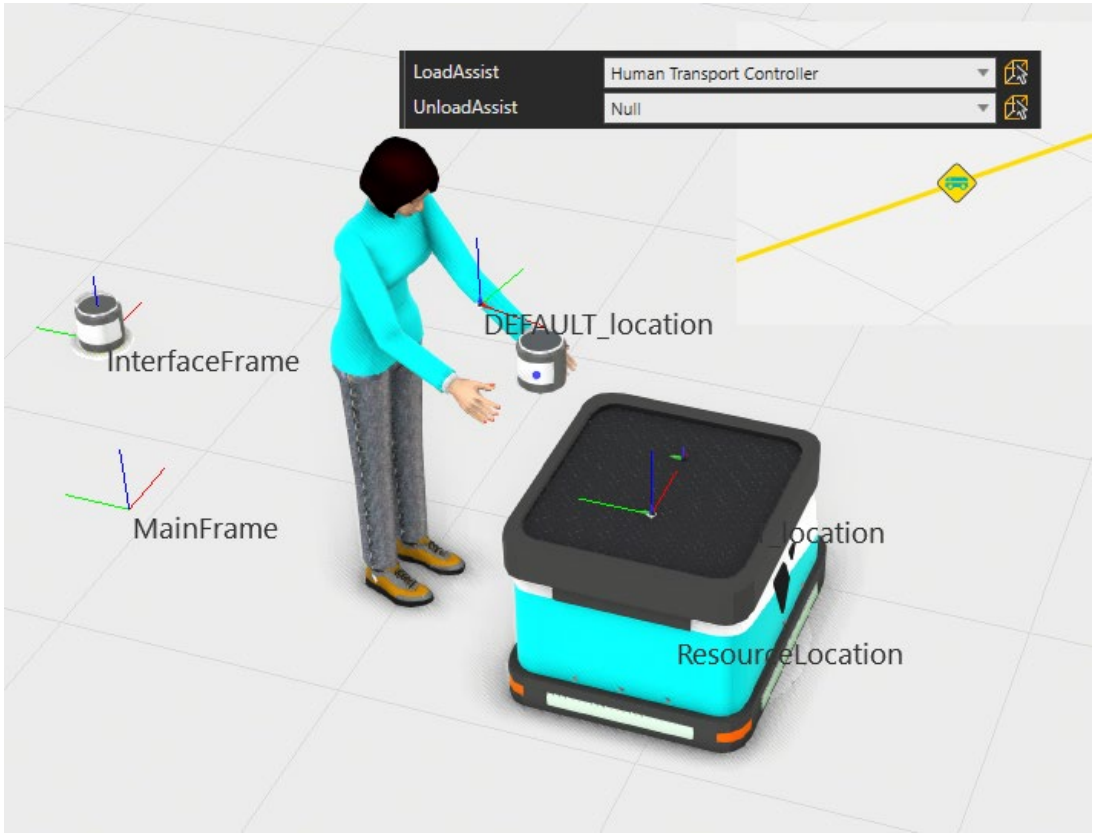
This way, it is possible to request a resource to bring and wait with a product next to the process. A usage example is shown in the picture below. In the example, the process requests for *Product #1*, and immediately after, the process waits another product (*Product #2*) to arrive from another link. After receiving *Product #2*, work is carried on that product to prepare it. After the work is finished, a resource possibly waiting with *Product #1* is allowed to place it into the process. Finally, the products are attached and shipped out as one.



LoadAssist and UnloadAssist

Another resource can load and unload products to the primary resource used for transportation. The used assistant is determined by the transport controller assigned to the transport link's property "LoadAssistant" or "UnloadAssistant". The property has a Null default value. Null is unassisted.

The primary resource will travel to the "ResourceLocation" specified in the corresponding process, and the assistant will be located between the ResourceLocation and the product unless the process is accompanied by the "AssistLocation" frame that will be used instead.

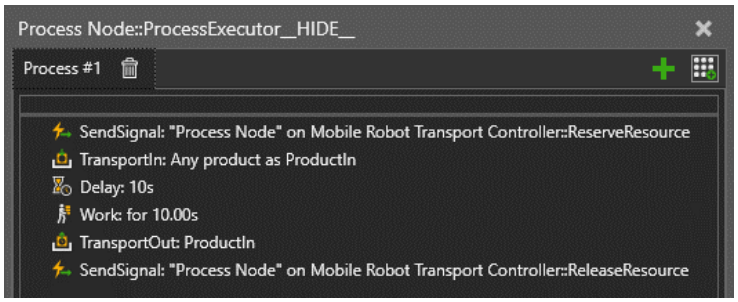


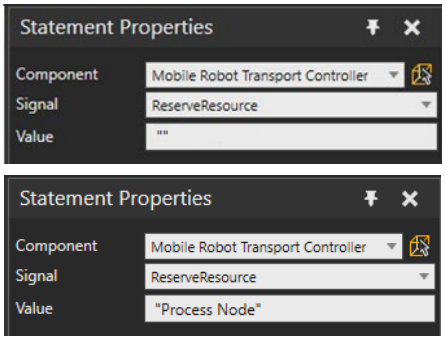
The request for assistance will be sent immediately upon departure to the loading or unloading location. The assistant will be waiting with the picked product if the assistance is ready before the primary resource and vice versa.

Note: If the assistant is attached to the primary resource, e.g., a robot arm on a robot controller is on-board, the request for assistance is sent after the primary resource is ready at the loading or unloading location and not upon departure like usually.

Reserving a resource to a process

If a resource must be dedicated to a process, it is possible to send a ReserveResource signal with Send Signal statement to the corresponding Transport Controller from a process.





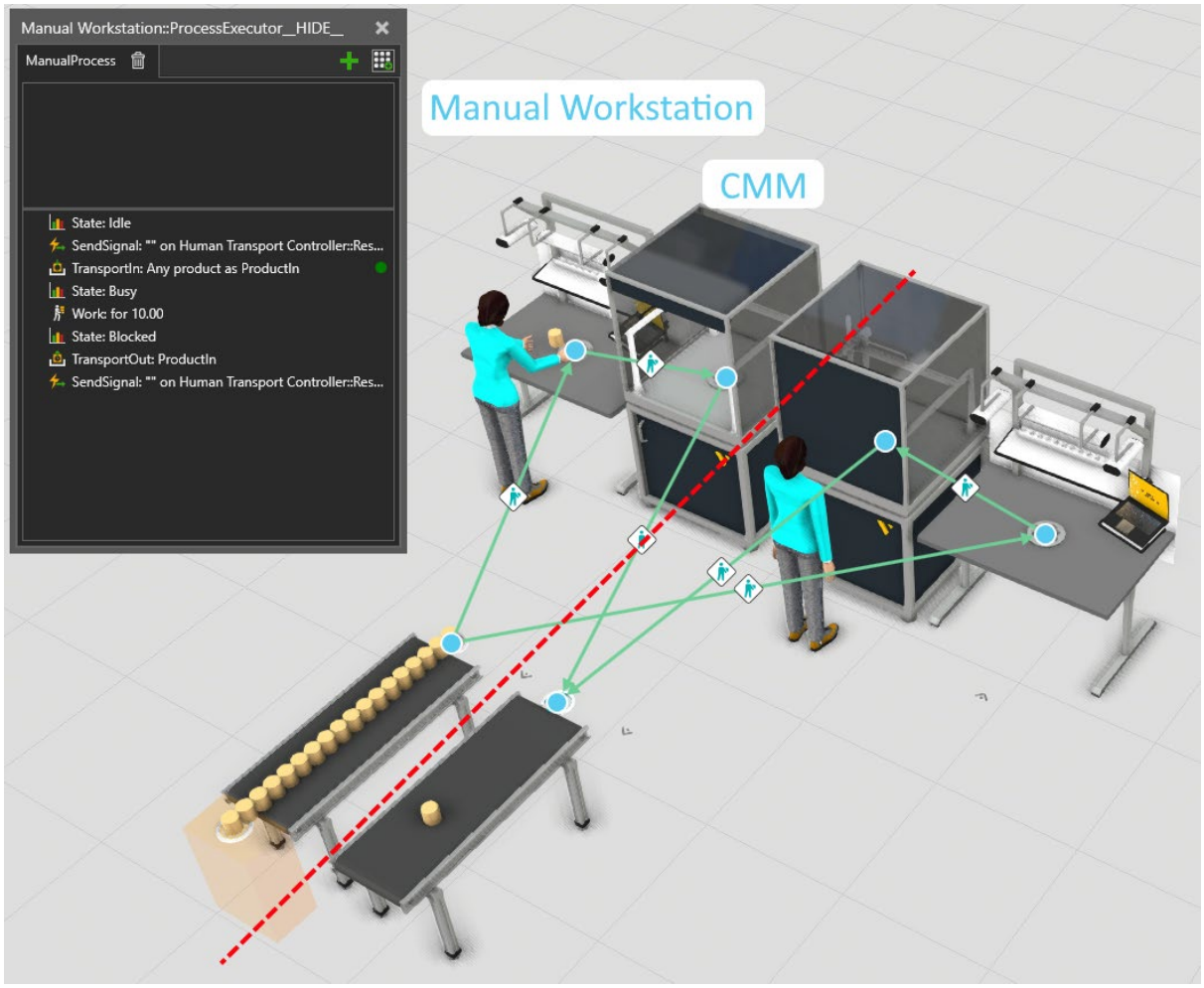
Once a Transport Controller receives the ReserveResource signal, it will reserve the next resource dispatched to transport products from/to or carry out work at the process. The value of the signal defines the process. If the value is an **empty string** (""), the caller process is used; otherwise, it must be the process **component's name** to which the reservation will be made.

When a resource is reserved for a process, no other resource (in the same controller) can be dispatched to it. The reserved resource will be dedicated to handle transport in/out and work tasks at the process until it is released or reserved by another process.

A resource can be reserved for multiple processes simultaneously. Therefore, ReleaseResource must be called at any point (after reserving) within the same process routine to release the process for other resources.

A typical example case is to reserve a mobile robot to wait for the next transport out after bringing products into a process. Another case is to have an operator dedicated to a sequence of tasks in a work cell so that any other operator cannot intervene.

In the picture below, the operators are working on “their own” side indicated by the red dashed line. ReserveResource and ReleaseResource are called in both the “Manual Workstation” and the “CMM”process on each side.



Navigation

Static obstacle avoidance

Once the simulation starts, a navigation mesh is generated. The navigation mesh will be marked with static obstacles that were detected during the mesh generation. The mesh is based on pathways, and if no pathways are connected to the Transport Controller, an invisible "global area" that covers the World floor can be used to detect and avoid static obstacles. This option can be toggled on or off from the "UseGlobalArea" property in the Transport Controller. Without the option, static obstacles are ignored.

A horizontal and vertical safety clearance will be added around the detected obstacles. The horizontal clearance is defined as the highest horizontal radius of connected resources plus the "ObstacleHorizontalClearance" value defined in the Transport Controller. The vertical clearance is the tallest resource plus the "ObstacleVerticalClearance".

Obstacles (e.g., light curtains) can be excluded by adding the components to the "ExcludeObstacles" component list. **Note:** A pathway has also a property for excluding all obstacles in that area.

By default, obstacles are detected based on their component bound box data. Some obstacles may be such a shape that the resource can go under it on some parts (a gate, for instance). In such a case, the obstacle can be added to "PrecisionObstacles" list. The geometries of a PrecisionObstacle are examined in more detail, which takes more time to process.

Dynamic obstacle avoidance

Dynamic avoidance can be enabled with the "Avoidance::Enabled" property in a Transport Controller. Dynamic avoidance is a global functionality in a layout. Enabling the avoidance will make all connected resources visible to other controllers and thus avoidable. If avoidance is disabled, the resources will not avoid each other and are ignored by other controllers.

Sensoring

Each resource will have a set of two sensors that will rotate and point to the travel direction. Both sensors are used to detect dynamic/moving obstacles only i.e., other resources. The first sensor is a “Slow Sensor” with a more extended range than the second sensor, a “Stop Sensor.”

The slow sensor is used to slow down the resource when an obstacle is detected within its range. The speed used is the “AvoidanceSpeed” as defined in the corresponding resource component. The stop sensor is used to stop the resource completely when an obstacle is detected in its range.

A fixed timer can be applied to the stop. If the timer is set, the resource will not initiate any actions before the timer has been run to the end. After the timer has expired, the resource will initiate a new scan with the sensors and proceed normally.

Resources are not capable of re-routing and bypassing each other. This means that in some cases, the resources are forced to go through each other e.g., head-to-head collisions. This applies only if

the “IgnoreDeadLocks” property is enabled. If the property is disabled, the resources are in a deadlock and will not move anymore.

Scanning frequency is dynamic and depends on two settings, “MaxSamplingInterval” and “MaxTravelInterval.” Scanning is guaranteed to happen whenever one of the thresholds is reached.

Avoidance properties in Transport Controller

Enabled: Enable avoidance for connected resources and make them visible to other resources in the World (requires that the other resources have avoidance enabled).

SlowSensorRange: Length of the sensor. Measured from the origin of the resource component.

SlowSensorSideClearance: Width of the sensor. Measured from the bounding box of the resource component.

StopSensorRange: Length of the sensor. Measured from the origin of the resource component.

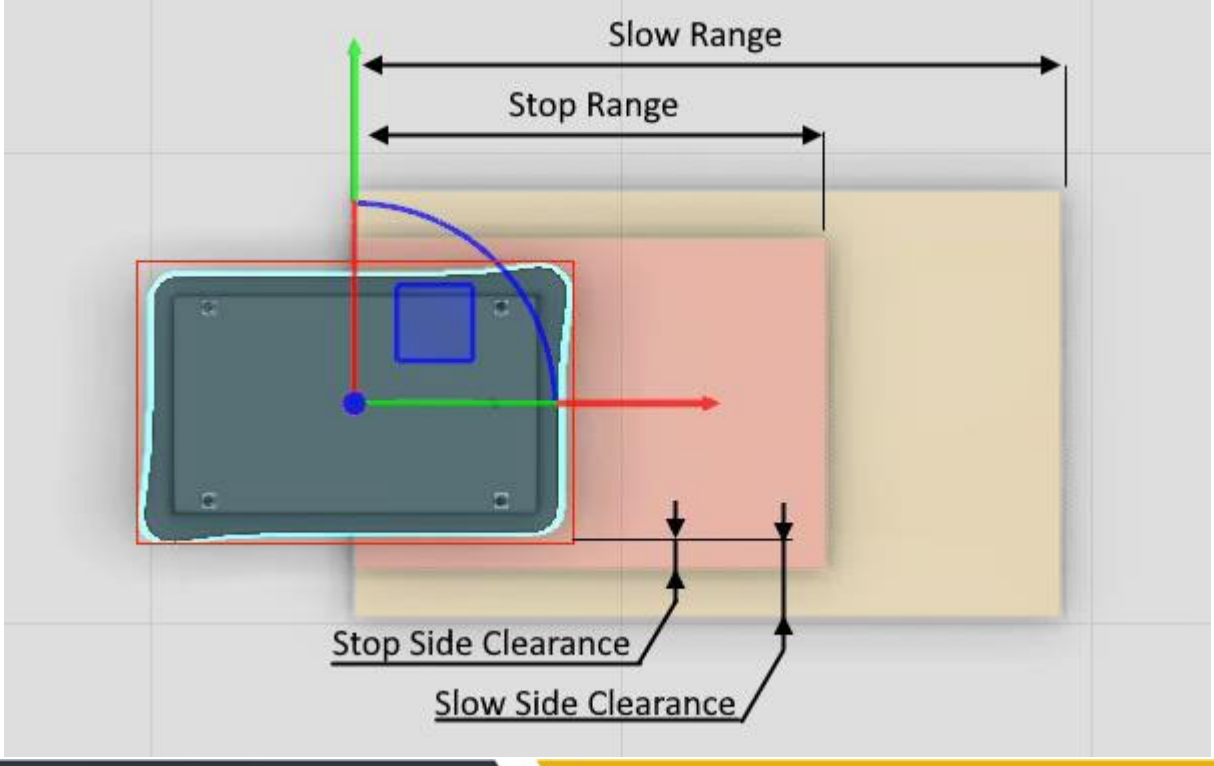
StopSensorSideClearance: Width of the sensor. Measured from the bounding box of the resource component.

StopTimer: Forced stop time when an obstacle is detected by the stop sensor.

MaxSamplingInterval: A maximum time interval between sensor scans.

MaxTravelInterval: A maximum travel distance between sensor scans.

IgnoreDeadLocks: Allows resources to go through each other when there are no options e.g. head-to-head collisions. If disabled, resources will be stopped for an infinite duration.



Pathways

There are two types of pathways in the eCatalog; Pathway Area and Pathway Curve.

Pathway Area

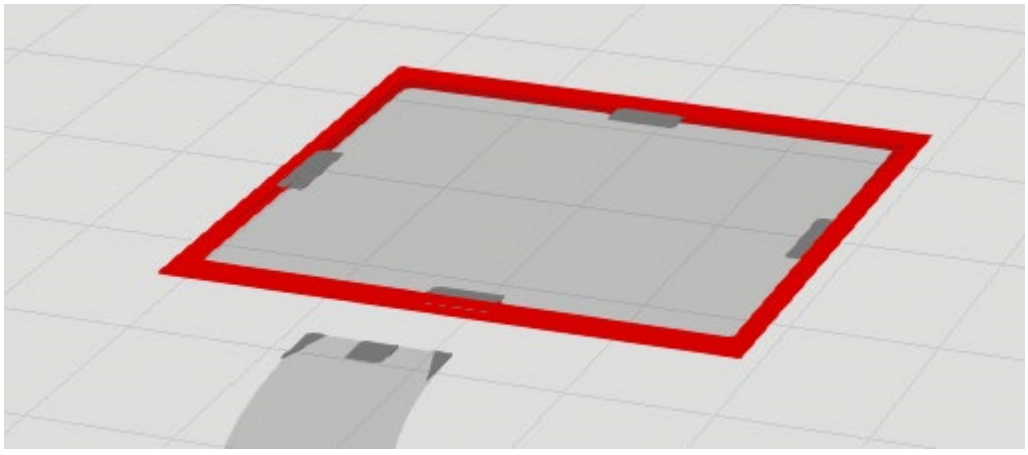
Pathway Curve

Pathways are used to generate the navigation mesh, that is used by the resources for travelling. Pathways can also be used to restrict access, or limit capacity or speed on the respective area.

A pathway must be connected to the Transport Controller to allow its connected resources to use and follow the pathways. Pathways are automatically connected if they are close enough to each other. Pathways can overlap but cannot be too far from each other. If an isolated pathway is not connected, a warning message is printed in the Output panel, once the simulation has been started:

> *[Human Transport Controller - WARNING]: Following areas are not connected to any other area: Pathway Area*

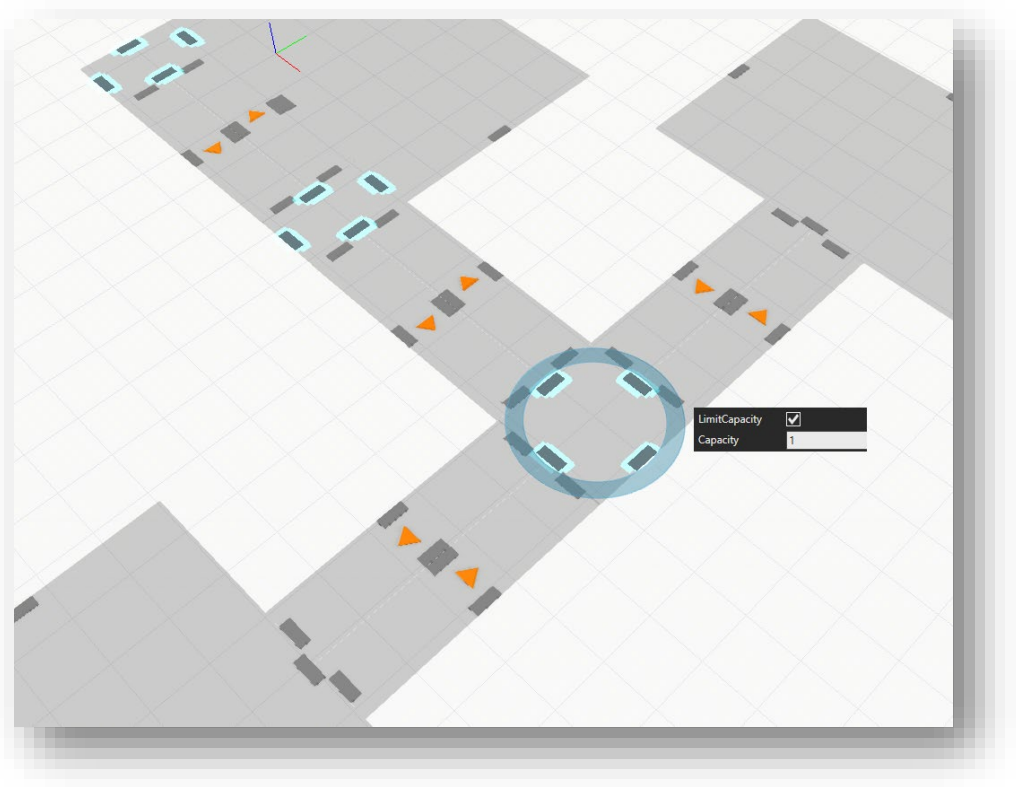
Also, the isolated area will be surrounded by a thick red line as shown in the figure below.



If no pathways are connected to the Transport Controller, and if the "UseGlobalArea" property (in the Transport Controller) is checked, an invisible "global area" covering the World floor with no limits is generated.

If “UseGlobalArea” is not checked and no pathways are connected, the resources ignore the navigation and go straight through the obstacles.

OneWay: Resources can be limited to move only in one direction along the path if enabled. With this option, it is possible to gain more control and construct lanes. When a junction of multiple lines is required, it is recommended to use a non-directional pathway area between, as shown in the figure below. One-way paths are always connected from the start and endpoints of the centerline along the defined direction.



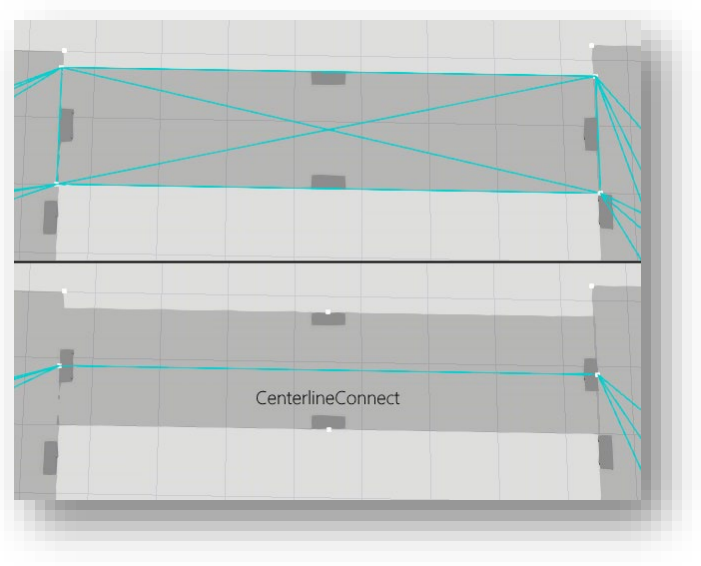
LimitSpeed: Can be used to limit the maximum speed that the resource can travel on the respective area. Note that the move speed properties of each resource also limits its maximum speed.

LimitCapacity: The capacity can be limited to avoid collisions. For example, the capacity can be set to one at the junctions, to allow only one resource to utilize the area at the time.

AddCost: Allows you to add extra distance in meters for calculating travel distances when reserving a resource by the controller, or when finding the suitable route by the resource. It can be used to avoid certain areas and to prioritize longer paths over shortcuts. The added cost does not affect any statistics of the actual travel distance.

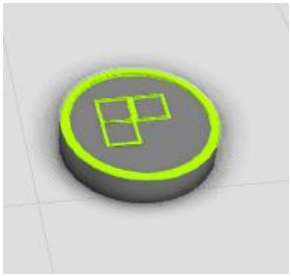
DetectObstacles: Obstacle detection can be turned on and off for each area.

CenterlineConnect: Controls if the connections are made from the corner points or centerline of the pathways. Pathway Curve and one-way paths are always connected from the start and endpoints of the centerline along the defined direction. See the figure below.



ShowResizeHandles: Shows or hides the dark grey handles on the edges of the pathways that allow resizing the pathways with the Interact -tool.

Pathway capacity grouping



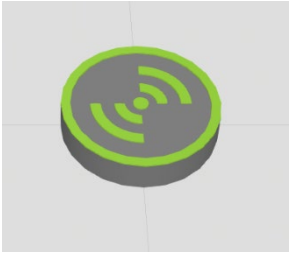
Pathway areas that have a capacity limit defined can be grouped by connecting the pathways via interface to the “Pathway Capacity Group” component. This allows “chaining of areas” which means that a resource will wait for available capacity in the group at the group entry.

CheckMode: allows you to define, which pathways in the area are considered in the capacity testing:

- Only the pathway areas on the incoming resource’s route must have capacity.
- All areas in the whole group must have capacity.

A Pathway can belong to only one group at a time.

Pathway area sensor

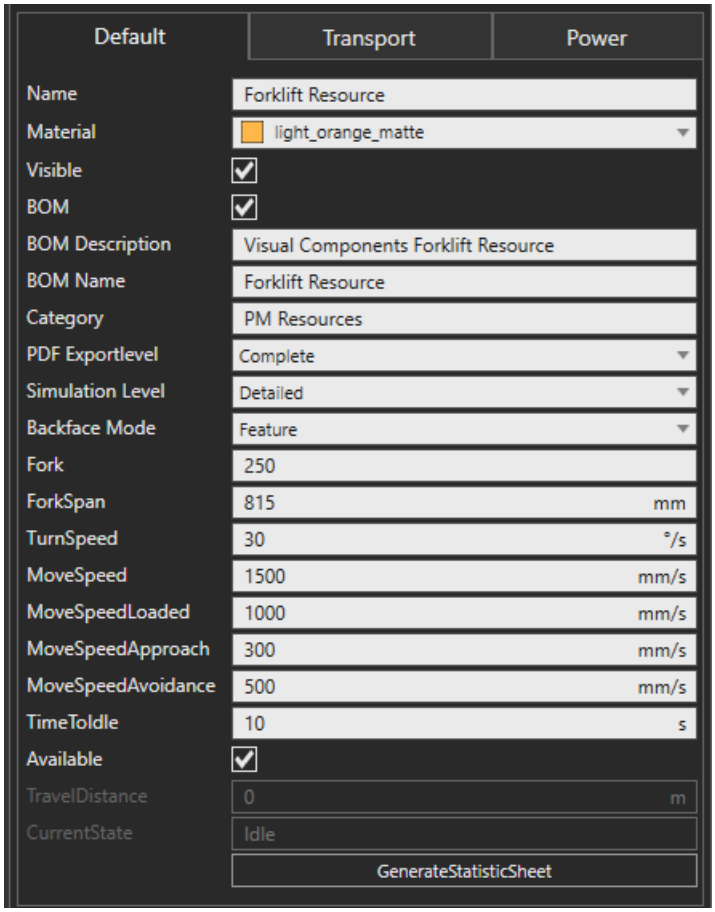


Pathway areas can be connected via interface to the “Pathway Area Sensor” component. This will make the sensor component detect if there are any resources in that area. You can then connect the “ResourceDetected” signal to any Boolean signal, for example to “IN_J1_Action” of “Rolling Steel Door v2”, which would make the door open when the sensor detects a resource entering the area, and close when the resource leaves the area.

Generic resource properties

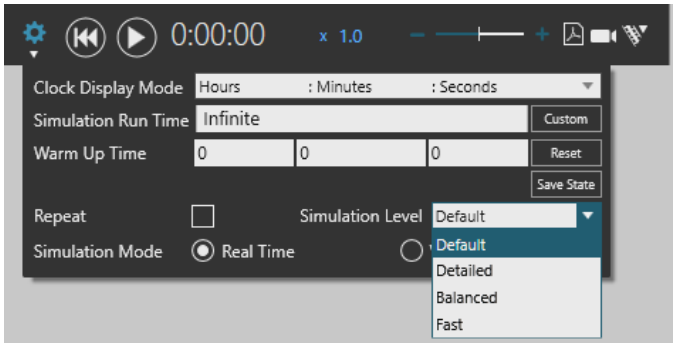
Humans, mobile robots, and forklifts share the same properties as described in this section.

Default tab



Simulation level: Change to "Fast" in order to improve performance and skip some visualization details (does not affect the simulation results).

Tip: Set globally for all resources under the simulation controls.



TurnSpeed: Specifies the maximum angular velocity when rotating in place.

MoveSpeed: Specifies the maximum velocity without payload.

MoveSpeedLoaded: Specifies the maximum velocity with a payload.

MoveSpeedApproach: Specifies the velocity used when approaching a resource position and reversing out.

MoveSpeedAvoidance: Specifies the velocity used when other resources are nearby.

TimeToIdle: See section Idling and Charging.

Available: Can be used to send the resource for a break during simulation. Once set to False, the resource will:

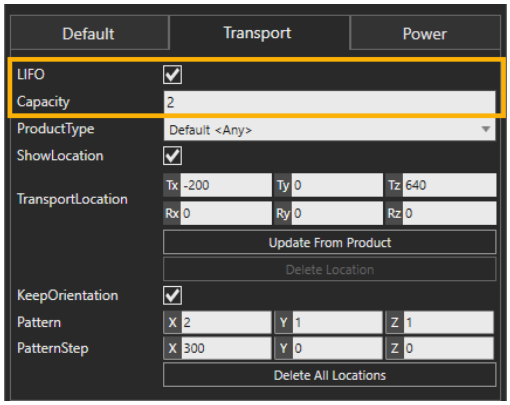
- Complete any existing tasks.
- Reject any new tasks.
- Change the state to “Break”.

The resource will continue accepting new tasks after Available is set to True.

TravelDistance: Total distance travelled during the simulation.

Transport tab

The transport tab defines the order of delivery (LIFO/FIFO[1]), how many products can be carried simultaneously, where the products are placed on the resource, and how they are handled.



The highlighted properties are applied globally to all products. Other properties are applied to each product type that is handled.

Global properties

LIFO: If enabled, the product last collected is delivered to its destination first.

Capacity: Total capacity of products on-board simultaneously

Product type-specific properties:

Product Type: A drop-down menu for selecting the product type for which the location and pattern are applied. The "Default <Any>" is applied to all product types that have no specific location defined. Select the last item "Add New…." on the menu to define a new location for a specific product type.

ShowLocation: Displays the location and pattern in the 3D World.

TransportLocation: An offset from the resource component origin used as the first position and reference for the pattern. Any change in the fields is saved immediately to the location.

Update From Product: Click to read and write the position of a product on-board (when the simulation is paused) to the **TransportLocation** field. If "Default <Any>" is selected, the first product on-board is selected regardless of its type. Otherwise, the targeted product type is selected.

KeepOrientation: If enabled, only a translation of the product to the corresponding location is applied when collecting (pick) and delivering (place). In other words, the orientation of the **TransportLocation** and the receiving process' ProductPosition is ignored.

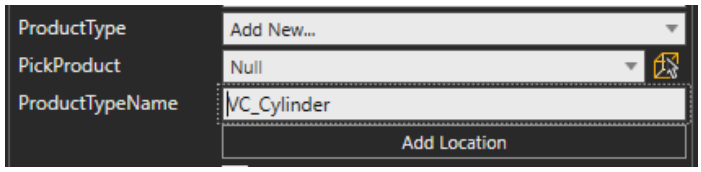
Pattern: Number of products along each axis defined by the TransportLocation.

PatternStep: Distance between the products along each axis.

[1] LIFO (Last-In-First-Out) and FIFO (First-In-First-Out) principles

Adding a new location

It is possible to type in the name of a product type (defined in Process tab/Products) to **ProductType** manually and then click **Add Location** to create a new location with a zero offset.



Alternatively, the simulation can be run until the desired product is on-board, then pause and pick the product with **PickProduct** that will read its type and location. Then click **Add Location**. If a location for the same product type already exists, it will not be overwritten. Instead, that product type will be selected as the **ProductType**.

Power tab

The Power tab is available for all vehicles (mobile robots, AGVs and forklifts), and it has at least one property “Enabled”. All other properties are shown if Enabled is set to True. ToChargeLimit and ChargeUntilLimit are a percentage of the (maximum) Capacity. All other units have been left out on purpose and can be in any arbitrary unit, e.g., Ah, kWh, kg, liters, etc.

Enabled: The resource takes power status into account if enabled.

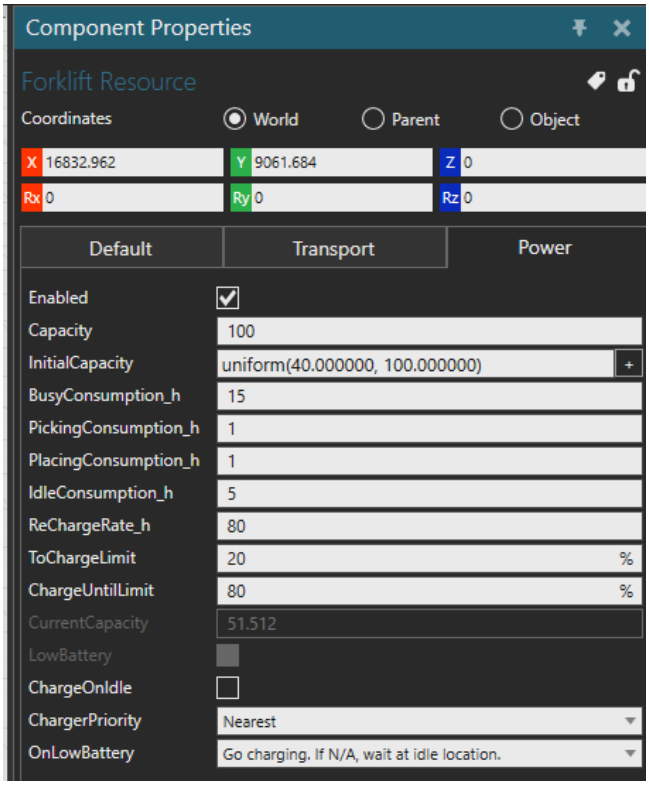
Capacity: The total maximum capacity.

InitialCapacity: Available capacity at the start of the simulation.

BusyConsumption_h: Defines the number of units per hour that the resource consumes when Busy.

PickingConsumption_h: Defines the number of units per hour that the resource consumes when Picking. For example, when a mobile robot is running an integrated conveyor.

PlacingConsumption_h: Defines the number of units per hour that the resource consumes when Placing. For example, when a mobile robot is running an integrated conveyor.



IdleConsumption_h: Defines the number of units per hour that the resource consumes when Idle (standby).

ReChargeRate_h: Charging rate as units per hour.

ToChargeLimit: Percentage of the total Capacity below which charging is required. Charging is initiated after existing tasks are completed. If no charging locations are available in the World, charging will be done in-place. See also section Idling and Charging.

ChargeUntilLimit: Percentage of the total Capacity below which leaving the charging station is not permitted. When the charge has reached this value, new tasks can be accepted. See also section Idling and Charging.

CurrentCapacity: Indicates the available capacity.

LowBattery: Indicates if the charging is required.

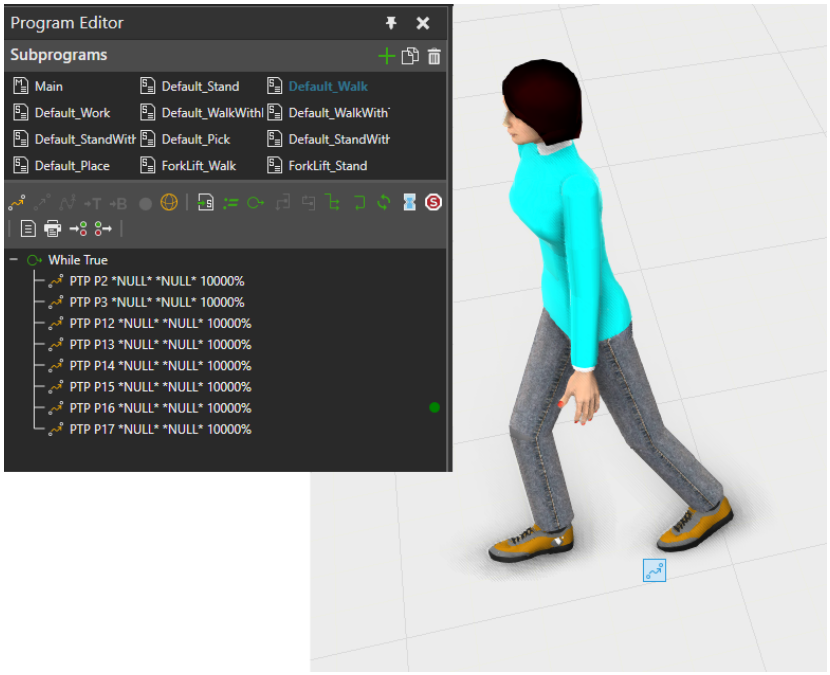
ChargeOnIdle: Option to go charging when on idle.

ChargerPriority: The resources can always go to Nearest charging station, or user can define the priority of the charging stations either locally in the resource or globally for all resources in the transport controller.

OnLowBattery: Defines if the resource stands still or goes to wait at an idle location, when the battery is low, but all the charging locations are occupied.

Human animations

The Human resource has default animations defined for walking with or without a product or a tool. The same applies to standing, picking, placing, and working. Animations are defined as robot program routines, and the defaults are used if no custom animations are found for the specific action.



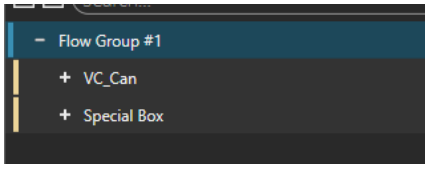
Custom animations can be created with a naming convention. The naming can be based on the carried product's product type, the tool component name, or the process name, depending on the active action.

Picking, placing, and transporting

Without a tool, picking, placing, and carrying can be animated based on the product type name. The following four routines are used:

- <product type name>**_Pick**
- <product type name>**_Place**
- <product type name>**_Walk**
- <product type name>**_Stand**

Where <product type name> is the required prefix. E.g., animation for a specific product type "Special Can" must be named with a prefix "Special Box": *"Special Box_Pick", "Special Box_Place"* etc.



Note: If multiple products are on-board, only the first product picked is considered when the walk and stand animations are selected.

With a tool, the product type name is ignored. The used tool's component name is used to define the animation. Pick and Place animations are not used with a tool.

- <tool name>**_Walk**
- <tool name>**_Stand**

In case multiple tools for the same task are available, the tool name can be given partially as a lookup name. For example, multiple pallet jacks can be available, but each component must have a unique name: "*Pallet Jack #1*", "*Pallet Jack #2*" etc. Then the animation prefix can be the common part of the names. In this case, leaving out the running number: "*Pallet Jack_Walk*".

Note: Animations are not time-scaled. If the duration of animation is longer than *PickTime*, *PlaceTime,* or *ProcessTime* (working), then it will be cut once the time is exceeded.

Tip: It is recommended to copy the default routines such as the "*Default_Walk*" and use touch up to update the upper body positions to each statement.

Working

Without a tool, work animation can be based on the process name where the work is being carried out. The prefix is the name of the process:

- <process name>**_Work**

For example: *"Process #1_Work".*

With a tool, the process name is ignored, and the tool component's name is used instead:

- <tool name>**_Work**

For example: "*Power Drill_Work".*