Possible values of **Command** enumeration which will be included in the **WriteGoTo, PickCart, PlaceCart, QueryState, QueryStateValueReturn, Cancel** events:

**GOTO**, **PLACE**, **PICK.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request Event** | **Function Description** | **Inputs Parameters** | **Response**  **Event** | **Notes** |
| **~~CheckRobotReadyRequest~~** | ~~After robot behaviour is linitialized, it needs to check if the ROS Edge Node initiialization is done or not.~~ | ~~robotID,~~ | **~~RobotReadyResponse~~** | ~~Robot behaviour needs to check if a robot is ready or not, then go to query the warehouse backend~~ |
| **~~RobotReady~~**  **~~Response~~** | ~~Response to~~ **~~CheckRobotReadyRequest.~~** ~~ROS Edge Node report its availability after it’s initialized for connecting with robot~~ | ~~robotID,~~  ~~IsReady~~ *~~(boolean)~~* | **~~/~~** | ~~After receiving this event, Robot Behaviour can start to ask the warehouse backend component for a new Pickup task including the picking and storage positions for the first iteration.~~  ~~Currently this event is used by myself to manually implement the simple robot behaviour, because I run the ROS Edge Node together with this robot behaviour at same time, this event is sent by ROS Edge Node to robot behaviour when its initialization phase is done.~~ |
| **RobotReadyBroadcast** | ROS Edge Node automatically report its availability to robot behavior after it has connected to robot | RobotID,  RobotIP,  IsReady (boolean, default is true) |  | After receiving this event, Robot Behaviour can config its robotID and its filter of smart beahvior to receive the events only with the specific robotID, then start to ask for a new picking point including the picking position for the first iteration.  The **‘isReady’** properties could be kept with default **‘true’** value by robot behaviour, when there is a ‘**QueryStateValueReturn**’ response from ROS Edge Node with the **‘CurrentState’** property equal to **‘unknown’,** which means something wrong when executing the current action, then robot behaviour could stop. |
| **WriteGoTo** | Send command to robot with the coordinate.  The 'command' field is fixed | robotID,  command = ”GOTO”,  coordinate, | **QueryStateValueReturn** | Its ‘command’ field is used for matching with the ‘command’ field of  **QueryStateValueReturn** received by Robot Behaviour, who will know the QueryStateValueReturn is the response of which robot command **(GoTo, PickCart, PlaceCart, QueryState, Cancel)** |
| **PickCart** | Send command to pick a cart with the info of scanned marker ID of the cart  The 'command' field is fixed | robotID,  command =” PICK”,  **~~markerID~~** | Same with ‘WriteGoTo’ | Same with ‘WriteGoTo’  **Note: the property ‘markerID’ will be used only in simulation ENV by ROS edge node to check the cart name to be send to ros service, this field will be removed in real robot, and all PickCart events are same.**  **To simplify development of the robot behaviour, robot behaviour can always use this property in the `PickCart` event in both Simulation and Real robot, ROS Edge Node will ignore it when it’s not used** |
| **PlaceCart** | Command robot to place the cart  The 'command' field is fixed | robotID,  command = ”PLACE” | Same with ‘WriteGoTo’ | Same with ‘WriteGoTo’ |
| **QueryStateValueReturn** | Response to the  **'GoTo'**,  **'PickCart'**, **'PlaceCart'**,  **‘QueryState’, ‘Cancel’** events | robotID,  command,  CurrentState |  | 1. Enumerated values of CurrentState field:   * **finished** * **unknown** * **queued** * **running** * **paused**   2. After receiving the five action events, ROS Edge Node will command the robot, and continuously query the execution status for corresponding action.  Once it detects the state is changed to the **unknown/finished** state, this event will be automatically created and report to Robot Behaviour.  So, Robot Behaviour just need to wait for this event to check the execution status by checking its ‘command’ field with the request event previously sent  2. if a **'QueryState'** command is explicitly sent from Robot Behaviour, after receiving it, ROS Edge Node will also execute and reply with 'QueryStateValueReturn'. |
| **QueryState** | Check current event command execution is finished or not if Robot Behaviour wants to explicitly check the status, in generally it will just wait for the response of ROS Edge Node | robotID,  command | **QueryStateValueReturn** | Same with ‘WriteGoTo’ |
| **CheckMarker** | Command robot to scan the QR Code seen by its camera | robotID | **MarkerReturn** |  |
| **MarkerReturn** | Response to **‘CheckMarker’** with the marker ID contained in QR Code of any objects (e.g. Carts, Door) scanned by robot camera. | robotID,  markerID |  | Through the returned marker ID, Robot Behaviour knows what object it is in front of the robot now.   * If cart, send **PickCart** * If Door, command the **door edge node** to open door |
| **CheckAvailability** | The event to check the overall availability of robot about whether there is any command is running, and including the current coordinate of robot | robotID |  |  |
| **AvailabilityReturn** | Response to  **CheckAvailability** | robotID,  OperationState,  NavigationType,  currentPosition |  | 1. Enumerated values of OperationState:   * **idle** * **moving**   2. Enumerated values of NavigationType:  “GoToComponent” | “PickComponent” | “PlaceComponent” |  “None”  3. currentPosition field contains the current coordinate of the robot |
| **Cancel** | Cancel the current running command | robotID,  command | **QueryStateValueReturn** | Same with ‘WriteGoTo’ |
| **Events between Robot Behaviour and Warehouse Backend** | | | | |
| **DockingRequest** | Request for a new docking pose according to **robot ID** which is the primary key in **Dock\_Point Table** | robotID,  robotIP | **DockingResponse** | Robot behaviour sends robot ID and robotIP to warehouse backend to query the docking point from the **Dock\_Points Table**, each robot has a fixed docking point according to its IP address |
| **DockingResponse** | Response to  **DockingRequest** | robotID,  hasNewPoint(boolean),  dockingPoint,  dockAuxliaryPoint, |  | 1. The **dockAuxliaryPoint** field isfixed point for a robot in the **storage side** in front of Door on the way to **docking area**. Robot will stop here. Robot behaviour will send ‘CheckMarker’ event to check the door status. Then open the door if it’s closed, and send new **WriteGoTo** event if door is open. Similar meaning with the storageAuxliaryPoint  2. **hasNewPoint** field is of ‘Boolean’ type. After receiving this event, robot behaviour will check the **‘hasNewPoint’** field:   * If **True**: starts to go back to docking area. Default value. * If **False**: no docking point for this robot. (In generally, this will not happen, because each robot will have its own docking point)   After checking this field is true, robot behaviour starts to command robot to go back to docking area |
| **NewPickPoint**  **Request** | Robot Behaviour ask warehouse backend for a new picking point | robotID | **NewPickPointResponse** | warehouse backend will manage the shared tables, and return new task for next iteration |
| **NewPickPoint Response** | Response to ‘**NewPickPointRequest**’ event. warehouse backend replies with a new picking point, where a new cart to be moved | robotID,  hasNewPoint(boolean),  pickPoint |  | 2. **hasNewPoint** field is of ‘Boolean’ type. After receiving this event, robot behaviour will check the **‘hasNewPoint’** field:   * If **True**: starts a new iteration for moving cart. * If **False**: no more cart to be moved |
| **NewStoragePointRequest** | Robot behaviour ask for warehouse backend for the storage point with the cart marker ID | robotID,  markerID | **NewStoragePoint**  **Response** | cart marker ID, is used to identify the Place position. |
| **NewStoragePointResponse** | According to the cart marker ID, get the storage point | robotID,  markerID,  hasNewPoint (boolean),  storagePoint,  storageAuxliaryPoint |  | 1. The **storageAuxliaryPoint** field is fixed point for a specific point in the **picking side** in front of Door on the way to **storage area**. Robot will stop here and check door marker ID, if door is found(marker ID=1), it query door status. Then send new **GoTo** event if door is open.  2. **hasNewPoint** field is of ‘Boolean’ type. After receiving this event, robot behaviour will check the **‘hasNewPoint’** field:   * If **True**: starts to move cart to storage point. * If **False**: no more cart to be moved |
| **NoCartNotice** | It’s sent when Robot behaviour received **MarkerReturn** response, but there’s not any scanned marker ID. It willNotices the warehouse backend there is not any cart at the picking point for updating the table property **'isAssigned'** from **True** to **False** | robotID,  pickPoint | **CartNoticeResponse** | This event is sent by robot behaviour.  After receiving this event, warehouse backend will update the **‘isAssigned’** from **true** to **false** for this picking piont.  After the value is changed to false, it means this place could be assigned to another robot, and a new cart may be put in this place |
| **CartMovedNotice** | Robot behaviour  Notices the warehouse backend the cart at the picking point has been moved. The backend will updat the table property **'isAssigned'** from **True** to **False** | robotID,  pickPoint | **CartNoticeResponse** | This event is sent by robot behaviour after the **PlaceCart** event has been executed successfully, warehouse backend will update the **‘isAssigned’** from **true** to **false** for this picking piont |
| **CartNoticeResponse** | Response to  **NoCartNotice,**  **CartMovedNotice** | robotID,  noticeStatus = "OK" |  | After robot behaviour receives this event, send another  **NewPickupRequest** for next iteration |