Order API for Factory 4.0

This is version 1.1.1, all JSON is subject to change. I will notify you if it has changed.

Contact me on Slack or Email (barn1855@vandals.uidaho.edu) if you have any questions or want to request a change.

Confirmation:

Message Confirmation: JSON→ (“msg_type”: “message confirmation”, “msg_id”: “SO1000”, “msg_type_received”: “order”)

Each time the cloud or the simulator receives a message they should send back confirmation of receiving that message.

- ‘msg_type’ just shows that this is for confirmation.
- ‘msg_id’ is the ID that was generated for the messages that are being confirmed. For example, the ID here is ‘SO1000’, this is a ‘Send Order’ ID so the message being confirmed is the ‘Send Order’ described in the section below.
- ‘msg_type_received’ shows what type of message was just received.

[Cloud → Simulator]

All JSON packages sent from the cloud to the simulator require a msg_type, cloud_id.

- The ‘msg_type’ lets the simulator know what kind of data it is receiving.
- The ‘cloud_id’ is a unique ID generated on the cloud end for each message. This is used for message confirmation and for tracking down each message sent. The cloud and simulator database should keep track of all messages sent using this ID.

Send Order: JSON→ (“msg_type”: “order”, “cloud_id”: “SO1000”, “location”: “location01”)
This is a request to the simulator to order a disk from the warehouse, send it through the factory process, and then ship it out.

- ‘msg_type’ order.
- ‘cloud_id’ looks like ‘SO1000’ standing for ‘Sent Order’.
- ‘location’ is the position of the supplies in the HBW, example:
  |location01| |location02| |location03|
  |location04| |location05| |location06|
  |location07| |location08| |location09|

Each time an order is completed the simulator will send the cloud an order status, this is defined in the section [Simulator → Cloud].

Request Status: JSON→ (“msg_type”: “request_status”, “cloud_id”: “RS1000”)
At any point, while the simulator is running the cloud can request the status of the factory. When this request is made, both ‘Status’ and ‘Inventory’ will send back JSON data. The ‘Status’ is defined in the section [Simulator → Cloud]

- ‘msg_type’ request_order_status.
- ‘cloud_id’ looks like ‘RS1000’ standing for ‘Request Status’.

Perform Inventory: JSON→ (“msg_type”: “perform inventory”, “cloud_id”: “PI1000”)
This is a request to the simulator to perform an inventory. If the factory is not running then ‘Inventory Status’ will send an OKAY status, perform an inventory, then send the complete inventory to the cloud. If the factory is performing another operation then ‘Inventory Status’ will send an ERROR status and not perform an inventor.

- ‘msg_type’ perform inventory.
- ‘cloud_id’ looks like ‘PI1000’ standing for ‘Perform Inventory’.

Cancel Order: JSON → (“msg_type”: “cancel order”, “cloud_id”: “CO1000”)
If an order is still being processed and has not reached the oven yet, then the cloud can cancel the order. If the order is able to be canceled then ‘Cancel Status’ will send back an “order_canceled”: “True”. If not, then “order_canceled”: “False” and the order will complete.

- ‘msg_type’ cancel order.
- ‘cloud_id’ looks like ‘CO1000’ standing for ‘Cancel Order’.

Webcam: JSON → (“msg_type”: “webcam”, “cloud_id”: “WP1000”, “power”: “True”,)
This is used to activate or deactivate the webcam on the SSC.

- ‘msg_type’ webcam.
- ‘cloud_id’ looks like ‘WP1000’ standing for ‘Webcam’.
- ‘power’ True/False for is power is on or off.

Control Webcam: JSON → (“msg type”: “control webcam”, “cloud_id”: “CW1000”, “y_turntable”: 1, “x turntable”: 1)
Control the webcam via the cloud by sending the y-turntable a coordinate and the x-turntable a coordinate.

- ‘msg_type’ control webcam.
- ‘cloud_id’ looks like ‘CW1000’ standing for ‘Control Webcam’.
- ‘y_turntable’ this points the camera up or down.
- ‘x_turntable’ this points the camera left to right.

[Simulator → Cloud]

All JSON packages sent from the simulator will have a msg_type

- The ‘meg_type’ lets the cloud know what kind of data it is receiving.

- Notice that the ID's for the simulator messages are not unique. All message ID's are generated in on the cloud, the simulator is just reacting to those request so each message from the simulator will just have "cloud_id" instead of a unique ID.

Order Status: JSON→ ("msg_type": "order status", "cloud_id": "SO1000", "disk_color_id": "Red01" "order_complete": "True")
Once an order is complete, 'Order Status' will be sent to the cloud to notify if it was successful or not.

- 'msg_type' order status.
- 'cloud_id' is the ID generated on the cloud side, this ties the data to the request that generated it.
- 'disk_color_id' This ID has a color then a number, the number is to differentiate between the different disks of the same color.
- 'order_complete' indicates if the order was completed or not.

Status: JSON→ ("msg_type": "status", "cloud_id": "RS1000", "running": "True", "HBW": "True", "VGR": "False", "MPO": "False", "SSC": "False", "SLD": "False")
Status check on the warehouse, reports back if it is running and what modules are currently active.

- 'msg_type' status.
- 'cloud_id' is the ID generated on the cloud side, this ties the data to the request that generated it.
- 'running' True/False for factory activity.
- 'HBW' True/False for HBW activity.
- 'VGR' True/False for VGR activity.
- 'MPO' True/False for MPO activity.
- 'SSC' True/False for SSC activity.
- 'SLD' True/False for SLD activity.

Inventory: JSON→ ("msg type": "inventory", "cloud_id": "PI1000",
"location01": "RED01", "disk_stored": "True", "pallet_stored": "True",
"location02": "RED02", "disk_stored": "True", "pallet_stored": "True",
"location03": "RED03", "disk_stored": "True", "pallet_stored": "True",
"location04": "BLUE01", "disk_stored": "True", "pallet_stored": "True",
"location05": "BLUE02", "disk_stored": "True", "pallet_stored": "True",
"location06": "BLUE03", "disk_stored": "True", "pallet_stored": "True",
"location07": "WHITE01", "disk_stored": "True", "pallet_stored": "True",
"location08": "WHITE02", "disk_stored": "True", "pallet_stored": "True",
"location09": "WHITE03", "disk_stored": "True", "pallet_stored": "True")
Shows all the HBW inventory data.

- 'msg_type' inventory.
- 'cloud_id' is the ID generated on the cloud side, this ties the data to the request that generated it.
- 'location' HBW locations with the Color ID of each disk:

  | RED01 || RED02 || RED03 |

  | BLUE01 || BLUE02 || BLUE03 |

| WHITE01 || WHITE02 || WHITE03 |

- ‘disk_stored’ True/False if the disk is stored in HBW.
- ‘pallet_stored’ True/False if the pallet is stored in HBW.

Cancel Status: JSON → (“msg_type”: “cancel status”, “cloud_id”: “CO1000”, “order_id”: “SO1000”, “canceled”: “True”)
When the cloud asks for an order to be canceled, this is the message that is sent back. It will let you know if the order was able to be canceled. What determines if an order can be canceled or not is if the supply disk has reached the oven on the MPO yet. Once a product is in the oven it is set to ship and can not be canceled.

- ‘msg_type’ cancel status.
- ‘cloud_id’ is the ID generated on the cloud side, this ties the data to the request that generated it.
- ‘order_id’ is the order ID generated on the cloud side.
- ‘canceled’ True/False if the order was able to be canceled.

Webcam Status: JSON → (“msg_type”: “webcam status” , “cloud_id”: “WP1000”, “power”: “True”, “y_turntable”: 1, “x_turntable”: 1)
When ‘Webcam’ or ‘Control Webcam’ sends a request the ‘Webcam Status’ will send back the current webcam status to show if any changes have been made.

- ‘msg_type’ webcam status.
- ‘cloud_id’ is the ID generated on the cloud side, this ties the data to the request that generated it.
- ‘power’ True/False for is power is on or off on the webcam.
- ‘y_turntable’ current location of the up and down axis.
- ‘x_turntable’ current location of the left to right axis.

Unable Status: JSON→ (“msg_type”: “unable status” , “cloud_id”: “PI####”,)

- ‘msg_type’ unable status.
- ‘cloud_id’ is the ID generated on the cloud side, this ties the data to the request that generated it.

Factory Module Times:

- **HBW**: from high-bay to conveyor (7 Seconds), moving box back to high-bay (7 Seconds)
- **VGR**: from HBW conveyor to MPO oven (10 Seconds), from SLD to ship-out (5 Seconds)
- **MPO**: from oven to SLD conveyor (8 Seconds + oven time), Red oven time (5 Seconds), Blue oven time (4 Seconds), White oven time (3 Seconds)
- **SLD**: from the color sensor to sorter (4 Seconds + color travel), White travel (0 Seconds), Red travel (1 Second), Blue travel (2 Seconds)