



35 Lammerstraat
Ghent, BE 9000
hello@optionyx.com

ALLOTCODA DOCUMENTATION

Updated: 25th October 2023

DIMENSIONER (IOS APP).....	2
How to set up.....	2
How to use.....	2
MASTER DATA STORE (API).....	5
How to set up.....	5
Prerequisites.....	5
Configure the service.....	5
Configure the Orion integration.....	6
How to use.....	7
Adding data.....	7
Querying data.....	8
MASTER DATA STORE (WEB).....	8
How to set up.....	9
Prerequisites.....	9
Configure the service.....	9
How to use.....	10
BIN PACKING (ORION).....	12
How to set up.....	12
Prerequisites.....	12
Configure the Orion integration.....	12
TRUCKLOAD BUILDING (WEB).....	15
How to set up.....	15
Prerequisites.....	15
Configure the service.....	15
How to use.....	16

DIMENSIONER (IOS APP)

How to set up

In order to set up the iOS app you need to:

1. Navigate into its settings
2. Select the server configuration option
3. Enter the IP address you'll be hosting your Docker environment on

How to use

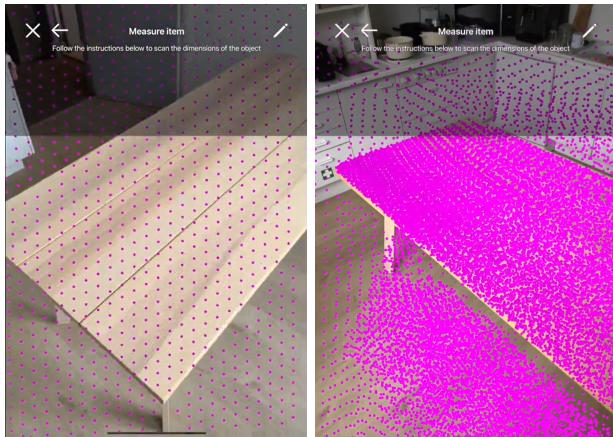
1. Select the “Scan” option to start collecting master data.



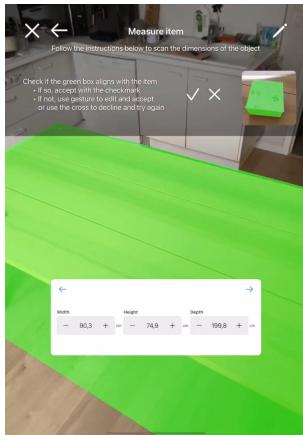
2. Line up your barcode to read it.



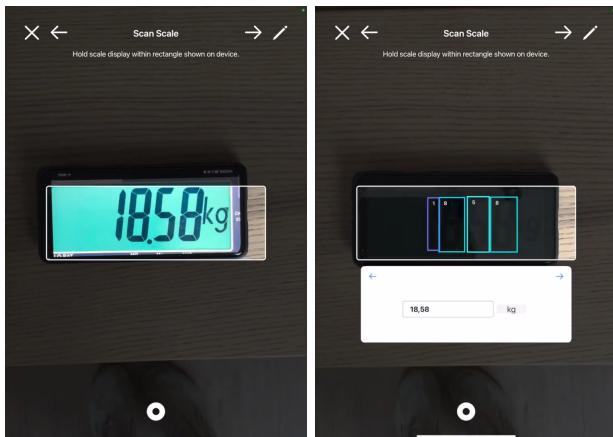
3. Tap the object you wish to measure to start dimensioning it. Tap again to stop.



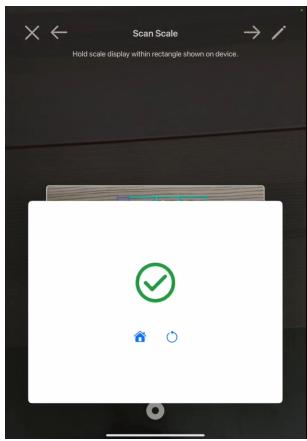
4. Verify that the green bounding box covers the object correctly, and adjust if necessary. You can always modify the dimensions manually.



5. Record the weight of an object by lining up with the scale's seven-segment display.



6. Finally, you can choose to go back to the home screen or start a new scan.



MASTER DATA STORE (API)

The Master Data Store is an API written with Python FastAPI that uses MongoDB for storing most of its data and an S3 bucket for storing images and other binaries.

How to set up

Prerequisites

- [mongo:4.0](#)
- [fiware/orion-lid:latest](#)
- [minio/minio:RELEASE.2020-08-18T19-41-00Z](#)

Configure the service

```
 1 master-data-store:
 2   build:
 3     context: ./master_data_store
 4   environment:
 5     - ORION_URL=http://orion:1026
 6     - MONGODB_URL=mongodb://mongo:27017
 7     - MINIO_URL=http://minio1:9000
 8     - MINIO_ACCESS_KEY=admin
 9     - MINIO_SECRET_KEY=password
10     - API_LOCATION=http://master-data-store
11   ports:
12     - "80:80"
13   depends_on:
14     - orion
15     - mongo
16     - minio1
17     - minio2
18     - minio3
19     - minio4
20   restart: on-failure
```

Environment variables

- ORION_URL*: the URL to your Orion service

-
- **MONGODB_URL***: the connection string to your MongoDB
 - **MINIO_URL***: the URL to your Minio service
 - **MINIO_ACCESS_KEY***: your Minio service access key
 - **MINIO_SECRET_KEY***: your Minio service secret key
 - **API_LOCATION***: the external URL on which this service is exposed

Configure the Orion integration

```
1 orion-mds-bridge:
2   build:
3     context: ./orion_mds_bridge
4   environment:
5     - ORION_URL=http://orion:1026
6     - MDS_URL=http://master-data-store
7   ports:
8     - "1029:1028"
9   depends_on:
10    - orion
11    - master-data-store
12   restart: on-failure
```

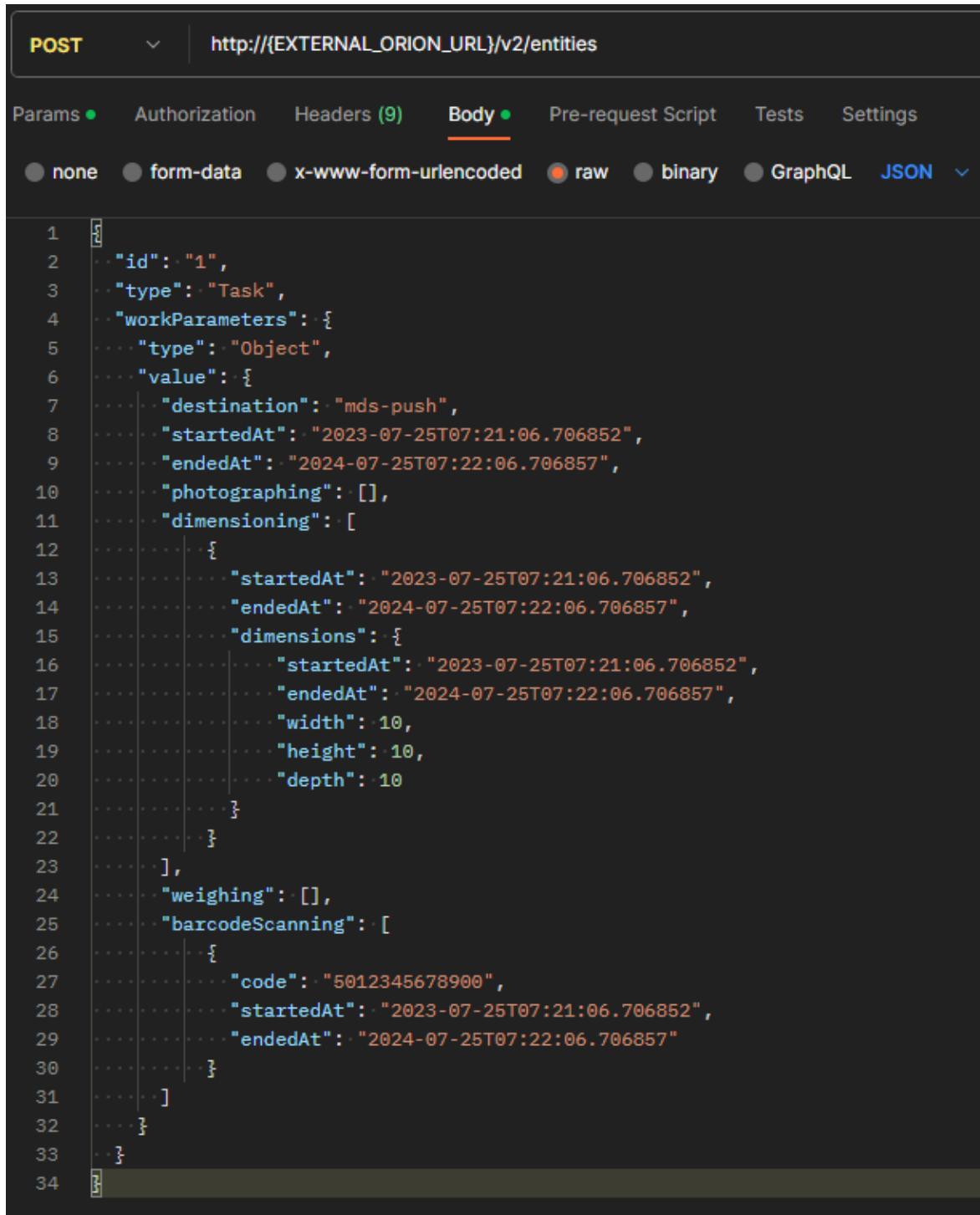
Environment variables

- **ORION_URL***: the URL to your Orion service
- **MDS_URL***: the URL to your Master Data Store service

How to use

Adding data

Use the “mds-push” destination, providing a workParameter with a value that fits the Item schema (see {API_LOCATION}/docs, /redoc, or /openapi.json for more information).



The screenshot shows the Postman application interface. A POST request is being made to the URL `http://{{EXTERNAL_ORION_URL}}/v2/entities`. The "Body" tab is selected, indicating the request type is "raw". The raw JSON payload is as follows:

```
1  {
2    "id": "1",
3    "type": "Task",
4    "workParameters": {
5      "type": "Object",
6      "value": {
7        "destination": "mds-push",
8        "startedAt": "2023-07-25T07:21:06.706852",
9        "endedAt": "2024-07-25T07:22:06.706857",
10       "photographing": [],
11       "dimensioning": [
12         {
13           "startedAt": "2023-07-25T07:21:06.706852",
14           "endedAt": "2024-07-25T07:22:06.706857",
15           "dimensions": {
16             "startedAt": "2023-07-25T07:21:06.706852",
17             "endedAt": "2024-07-25T07:22:06.706857",
18             "width": 10,
19             "height": 10,
20             "depth": 10
21           }
22         }
23       ],
24       "weighing": [],
25       "barcodeScanning": [
26         {
27           "code": "5012345678900",
28           "startedAt": "2023-07-25T07:21:06.706852",
29           "endedAt": "2024-07-25T07:22:06.706857"
30         }
31       ]
32     }
33   }
```

Querying data

Use the “mds-query” destination, providing a workParameter with a value that contains either a “dbId” field (an ObjectId string) or a “ean”¹ field (the value provided with the “barcodeScanning.code”).

The screenshot shows two identical Postman requests for querying data. Both requests are POST methods to `http://{EXTERNAL_ORION_URL}/v2/entities`. The "Body" tab is selected, showing JSON data. The JSON structure is as follows:

```
1 {  
2   ... "id": "2",  
3   ... "type": "Task",  
4   ... "workParameters": {  
5     ... "type": "Object",  
6     ... "value": {  
7       ... "destination": "mds-query",  
8       ... "dbId": "651fb620ee71bd712177fb2"  
9     }  
10    ... }  
11 }
```

The second request is identical to the first, except for the "dbId" value which is replaced by "ean": "5012345678900".

¹ean: European Article Number

MASTER DATA STORE (WEB)

The web module of the Master Data Store is built in React and offers a complete overview of the collected data. It retrieves data from the Master Data Store.

How to set up

Prerequisites

- [master-data-store](#)

Configure the service

```
1 master-data-store-web:
2   build:
3     context: ./master_data_store_web
4   environment:
5     - REACT_APP_OPENAPI_LOCATION=http://master-data-store
6     - REACT_APP_API_LOCATION=http://127.0.0.1
7   ports:
8     - "3011:3011"
9   depends_on:
10    - master-data-store
11   restart: on-failure
```

Environment variables

- REACT_APP_OPENAPI_LOCATION*: internal URL to the Master Data Store
- REACT_APP_API_LOCATION*: external URL to the Master Data Store

How to use

1. Click the "Search by code..." field.

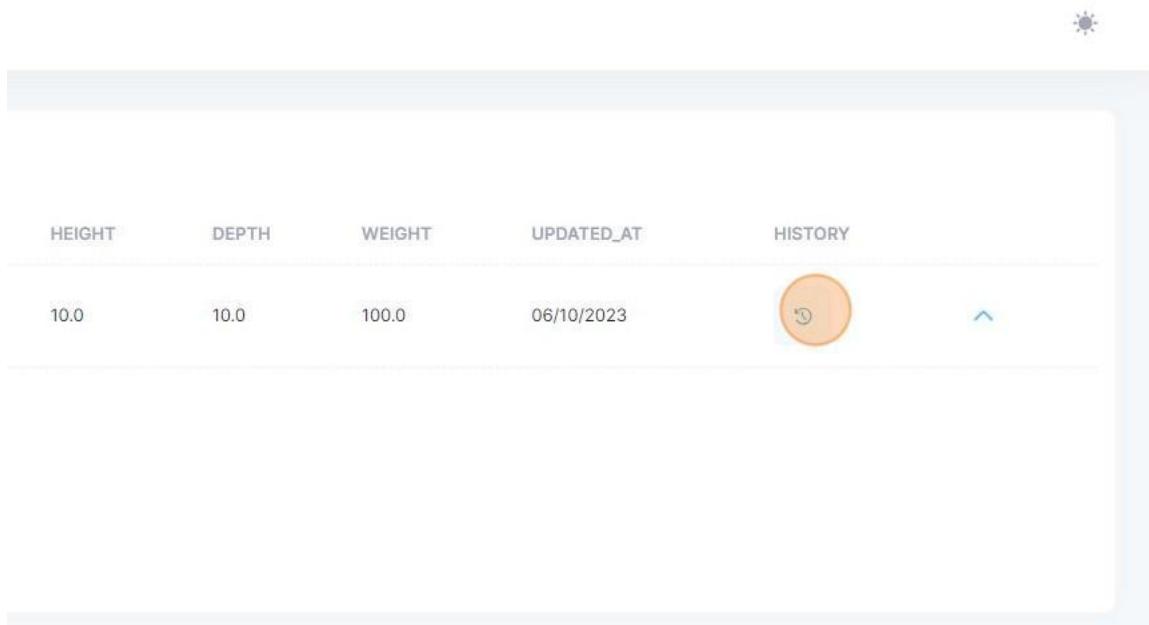
The screenshot shows a mobile application interface. At the top, there is a dark header with the word 'TRYX' and a back arrow icon. Below the header, the word 'Items' is displayed in a light-colored bar. The main area is titled 'Items'. In the center, there is a search bar with the placeholder text 'Search by code...'. To the right of the search bar is a large orange circular button. Below the search bar, there is a table with three columns: 'CODE', 'WIDTH', and two small blue icons. The table contains three rows of data:

CODE	WIDTH
5012345678900	10.0
651c39d7dc0a2fb7f7a2cbec	-
651c39d6dc0a2fb7f7a2cbeb	-

2. Type the barcode and press Enter
3. Open the collapsible to view details.

The screenshot shows a detailed view of an item. At the top, there is a small sun icon. Below it, there is a table with columns: HEIGHT, DEPTH, WEIGHT, UPDATED_AT, and HISTORY. The values are: 10.0, 10.0, 100.0, 06/10/2023, and a history icon. To the right of the HISTORY column is a large orange circular button with a downward arrow, indicating a collapsible section. Below this table, there is a large, mostly empty light blue rectangular area.

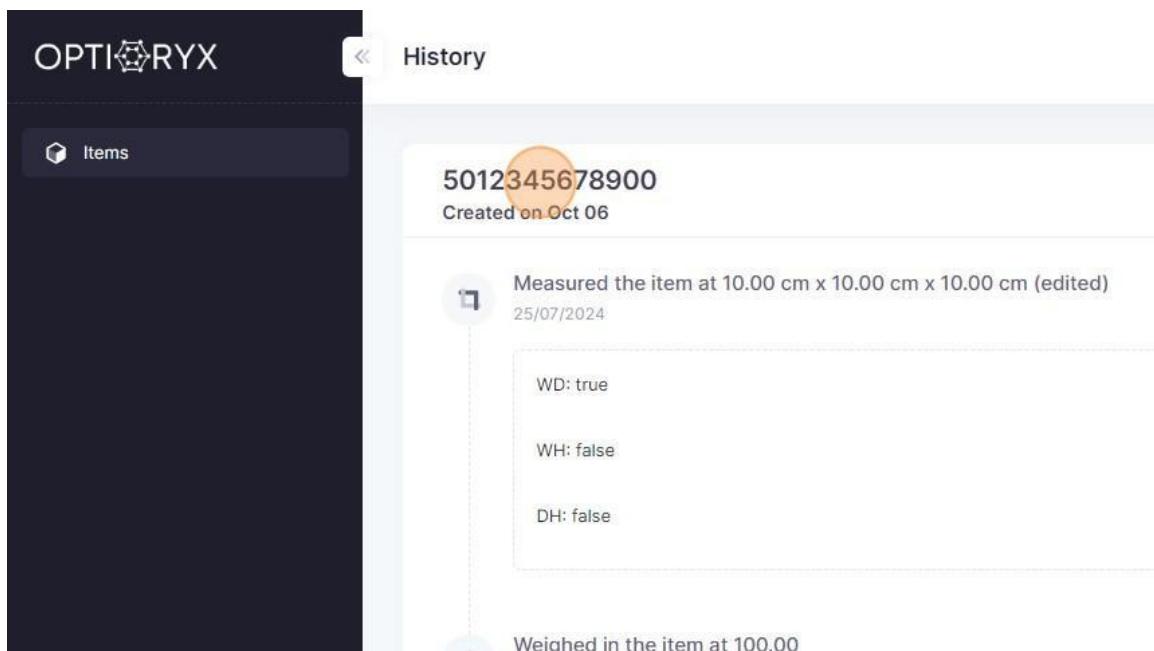
4. Click the history button to view the entire history for this item.



A screenshot of a mobile application interface. At the top, there is a blue header bar. Below it, a white card displays item details: HEIGHT 10.0, DEPTH 10.0, WEIGHT 100.0, and UPDATED_AT 06/10/2023. To the right of these details is a small orange circular icon containing a white circular arrow symbol, labeled 'HISTORY' above it. In the top right corner of the card, there is a small sun-like icon with radiating lines.

HEIGHT	DEPTH	WEIGHT	UPDATED_AT	HISTORY
10.0	10.0	100.0	06/10/2023	 

5. This will bring you to the history page.



BIN PACKING (ORION)

The Bin Packing Orion integration is a bin packing algorithm that uses data from the Master Data Store. Results from this algorithm are stored in a MongoDB collection, and can be viewed in the Truckload Building Web service. Additionally, step instructions can be generated optionally and viewed in the Adin webapp.

How to set up

Prerequisites

- [fiware/orion-lb:latest](#)

Configure the Orion integration

```
1 orion-bp-bridge:
2   build:
3     context: ./orion_bp_bridge
4   environment:
5     - ORION_URL=http://orion:1026
6     - MDS_URL=http://master-data-store
7     - API_KEY=012346789abcdef
8     - MONGODB_URL=mongodb://mongo:27017
9     - POSTGRES_USER=postgres
10    - POSTGRES_PASSWORD=password
11    - POSTGRES_DB=adin
12    - DB_HOST=adin-db
13    - MINIO_URL=http://minio1:9000
14    - BUCKET_URL=http://127.0.0.1:9001/optipackimages/
15    - MINIO_ACCESS_KEY=admin
16    - MINIO_SECRET_KEY=password
17   ports:
18     - "1018:1018"
19   depends_on:
20     - orion
21   restart: on-failure
```

Environment variables

- ORION_URL*: the URL to your Orion service

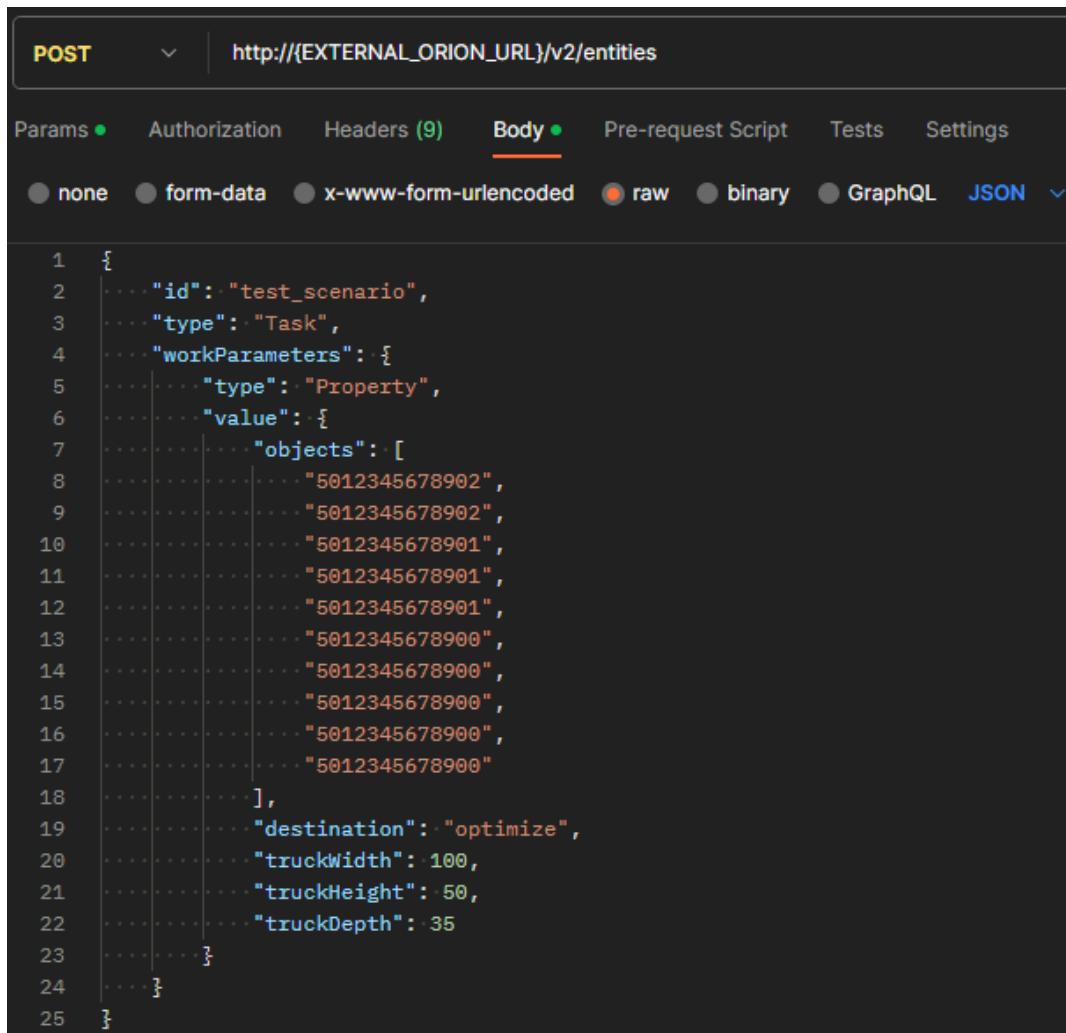
-
- MDS_URL*: the URL to your Master Data Store service
 - API_KEY*: your API key for the Optioryx API²
 - MONGODB_URL: the connection string to your MongoDB
 - POSTGRES_USER: the username for your Adin DB service
 - POSTGRES_PASSWORD: the password for your Adin DB service
 - DB_HOST: the name of your Adin DB service
 - MINIO_URL: the URL for your Minio service
 - BUCKET_URL: the URL to the Minio bucket on which to store the images
 - MINIO_ACCESS_KEY: your Minio service access key
 - MINIO_SECRET_KEY: your Minio service secret key

The latter portion of these variables are only required if you wish to view the packing results through Adin.

² Optioryx API: request your key at hello@optioryx.com

How to use

Use the “optimize” destination, providing a workParameter with the truck dimensions (“truckWidth”, “truckHeight”, “truckDepth”) and a value containing an array “objects” that contains the barcodes of the items to pack.



The screenshot shows the Postman interface with a POST request to `http://{EXTERNAL_ORION_URL}/v2/entities`. The **Body** tab is selected, showing the following JSON payload:

```
1 {  
2     "id": "test_scenario",  
3     "type": "Task",  
4     "workParameters": {  
5         "type": "Property",  
6         "value": {  
7             "objects": [  
8                 "5012345678902",  
9                 "5012345678902",  
10                "5012345678901",  
11                "5012345678901",  
12                "5012345678901",  
13                "5012345678900",  
14                "5012345678900",  
15                "5012345678900",  
16                "5012345678900",  
17                "5012345678900"  
18            ],  
19            "destination": "optimize",  
20            "truckWidth": 100,  
21            "truckHeight": 50,  
22            "truckDepth": 35  
23        }  
24    }  
25 }
```

TRUCKLOAD BUILDING (WEB)

The Truckload Building webapp is built in Next.js and offers an overview of the optimizations as well as a 3D visualization of the truckload builds. It retrieves optimizations from a MongoDB collection, populated with data processed by the Bin Packing Orion integration.

How to set up

Prerequisites

- [orion-bp-bridge](#)

Configure the service

```
1 truckload-building-web:
2   build:
3     context: ./truckload_building_web
4   environment:
5     - DATABASE_URL=mongodb://mongo:27017/test
6   ports:
7     - "3012:3000"
8   restart: on-failure
```

Environment variables

- DATABASE_URL: the connection string to your MongoDB

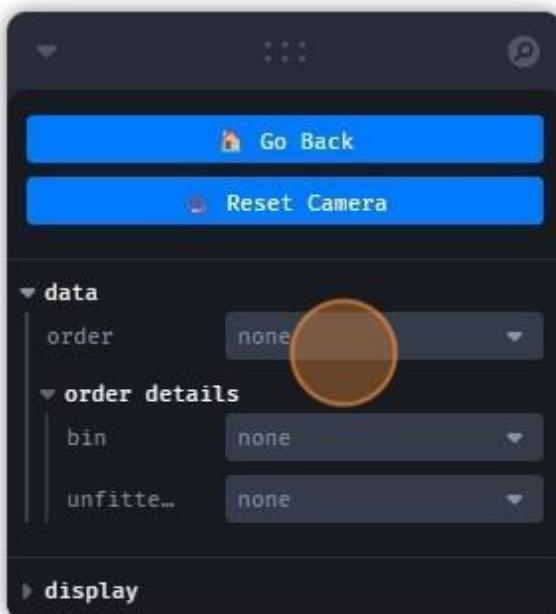
How to use

1. Select the optimization you wish to view.

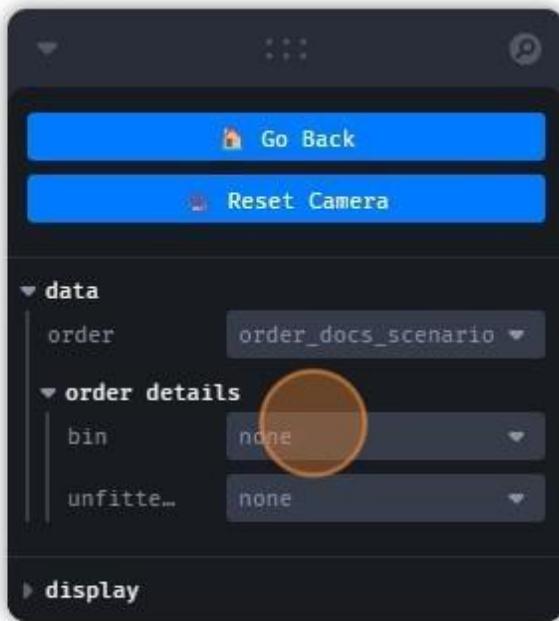


scenario ID	Description
test_scenario	Shop4CF
test_scenario	Shop4CF
docs_scenario	Shop4CF
docs_scenario	Shop4CF
docs_scenario	Shop4CF

2. Select an order.



3. Select a bin (truckload).



4. You can now view and interact with the visualization. Left-click while dragging to spin the truckload, right-click while dragging to pan the camera, scroll to zoom in and out.

