**Design For Creating a Moving list Based on CV x and y coordinates**   
  
**Purpose:**   
  
The primary objective of this document is to identify moving items based on the (x, y) coordinates provided by CV events, specifically where the item type is "placeholder." This list of moving items will subsequently be used to map them to the corresponding customers.

**Flow:**  
  
**Input:** Listening to Kafka messages coming from cv team with item type =” placeholder”  
**Ouput:** List containing all the Moving items**.**  
**Input:**  
**Example Response of CV Kafka event where item type = “placeholder”**  
"source":"isee"

"cart\_id": "123",

"sng\_did": "550e8400-e29b-41d4-a716-000000000222",

"event\_type": "add",

"Item\_type": "unknown",

"event\_uuid": "8e6381d5-63b6-4111-812b-066a322fc083",

"upc": "2903858848",## We are not getting this

"item\_confidence\_score": 0.9,

"cart\_event\_time": 1738161136799,

"cart\_confidence\_score": 0.9,

"location": "shelf1",

"create\_ts":1738161136810,

"date": "2025-01-29",

"store\_id": 8299,

"global\_id": "82990222",

"person\_global\_id": "82990001", \*\*\*NEW\*\*\*

"id": null,

"item\_event\_time": 1738161136799,

"person\_event\_time": 1738161136799,

"person\_confidence\_score": 0.9,

"track\_id": "1-130", \*\*\*NEW\*\*\*

"approach": 0, \*\*\*NEW\*\*\* 0: PERSON-FIRST, 1: CART-FIRST}

* CV also sends events regarding GlobalID movements, which contain (x\_cv, y\_cv) coordinates.

Based on the events received (item\_type = "placeholder") and GlobalID movements, identify (x\_cv, y\_cv) coordinates based on Global\_id and item\_event\_time received   
  
**Step 1: Convert CV Coordinates to RF Coordinates**

1. Convert CV (x\_cv, y\_cv) coordinates to RF (x\_rf, y\_rf) coordinates.

**Step 2: Identify the Region**

1. Determine the region based on the RF (x\_rf, y\_rf) coordinates.
2. Query the core server using the region ID to retrieve item details.

**Example Query:**

https://<hostname>/zebra/smartlens/items?$filter=region eq '4d03cf0b-f856-49f6-87fb-f8a40a10271d'

**Example Response:**

{  
 "id": "30340789004D0DC00001869F",  
 "x": 900,  
 "y": 900,  
 "z": 900,  
 "confidence": 60,  
 "floor": 0,  
 "type": "ITEM",  
 "region": "4d03cf0b-f856-49f6-87fb-f8a40a10271d",  
 "site": "936c1954-dce8-4783-90d3-cb9a7db19698",  
 "productId": "00123456789031",  
 "idFormat": "EPC",  
 "associatedId": "",  
 "associatedIdFormat": "",  
 "reason": "NONE",  
 "state": "VALID"  
}

**Step 3: Filter Events**

1. To reduce the amount of data retrieved from the core server:
   1. Ignore events with event type "Departure" and "Exit."
   2. Consider all remaining events as probably moving items.
2. **Option 1: Pagination**  
   When dealing with large volumes of item event data, pagination helps by processing data in smaller chunks—e.g., using LIMIT For instance, reading 10,000 rows at a time and filtering each batch improves performance and avoids loading the full dataset into memory. In next iteration we can read another 10,000 rows.
3. **Option 2: Time-Series Database**  
   A time-series database like InfluxDB or TimescaleDB is optimized for timestamped data, such as continuous streams of item movement events. It allows efficient filtering using time ranges (e.g., last 5 minutes) It allows real-time querying of only relevant, recent movement data—perfect for high-frequency tracking in retail environments.

**Step 4: Create List of Moving Items**

1. Compare the (x\_rf, y\_rf) trajectory of the items to (x\_cv, y\_cv) received.
2. If shelf\_id and item (x\_rf, y\_rf) match, then the item is said to be stationary.