Inventory API Overview

There are several important concepts to understand before interfacing with the Inventory APIs: best practices for handling large amounts of data, how inventory jobs are defined, and how segmentation tags are configured via API.

This guide is a general reference for using the APIs for inventory management including best practices, accessing job data, and how to set up granular fields and segmentation tags. See the Real-Time Inventory Service APIs guide for details about retrieving inventory levels instead.

> Note that if you are using both the user interface and the API, the "Product Code" field in the UI maps to the `upc` parameter in the API data.

Inventory API Best Practices

Due to the large amounts of data associated with inventory and the potential errors or slow performance when processing that data, there are some best practices that Kibo recommends for managing requests and responses.

Limiting Inquiry Responses

An inquiry request without a restriction on location may return a large number of results. The API can return up to n*100 results by default (where n = number of products in the request) but this may not be evenly split by 100 results per product. For instance, if there are two products then the response can include a maximum of 200 results; 50 results may be for one product while the other product has 150. Increase the value of the `limit` parameter in your inquiry request to change this setting if needed.

Sizing Adjust and Refresh Requests

Inventory Adjust and Refresh requests are placed into a queue of jobs that OMS processes one at a time. Due to this, the best practice is to submit fewer requests with more items per call, rather than numerous requests with a small number of items each. The Adjust API can accept up to a maximum of 1,000 items per call. The Refresh API can accept up to a maximum of 12,000 items per call. If these limits are reached, a Bad Request error will be returned.

Even though the Refresh API can support up to 12,000 items per call, Kibo recommends that you send requests in batches of 3,000 items for optimal performance.

Inventory Export Settings

You can use the Inventory API to configure your settings for the inventory export process to either S3 or FTP endpoints. There are three /create endpoints used to achieve this: a general endpoint, an S3-specific endpoint, and an FTP-specific endpoint. You can also configure fetch files with the Save Fetch Config API.

You may use any of these, but the general endpoint allows you to fine-tune your settings by specifying the location groups and sites that those configurations should apply to when inventory is being exported from those locations or sites. These specifications cannot be defined with the S3 and FTP specific endpoints.

> Configuring these settings will determine how these files are delivered and received, but that will not activate any exports. Contact to enable inventory imports and exports.

Deallocating Items

Sometimes, you may be stuck with items in "Allocated" status. These are not automatically cleared by a Refresh Inventory call, so you must make a Deallocate Inventory request. This is a two-step process because deallocation requires you to specify which shipments contain those line items.

1. First, make a GET Shipments query for shipments that contain the allocated item. In this example, the item is a product variation with the code 1234.

   ```
   GET /api/commerce/shipments/?filter=(items.variationProductCode=eq=1234) and shipmentStatus=ne=FULFILLED&fields=shipmentNumber,orderNumber,items.lineId,items.variationProductCode,items.quantity
   ```

2. Then you can use the shipment and order item IDs from the query response in the Deallocate call:

   ```
   [
    {
     "orderID": 15,
     "orderItemID": 1,
     "shipmentID": 25,
     "locationCode": "examplelocation",
     "upc": "1234",
     "quantity": 1
    }
   ]
   ```

Inventory Jobs

Once an inventory file has been uploaded, call the Get Job API to check the status of the inventory upload and see when it has been successfully processed, as well as the number of items. If you don't know the Job ID, the general Get Jobs API will return all jobs in the current queue.

```
https://t{tenantId}.{host}/api/commerce/inventory/v1/queue/{Job ID}
```

This API can be filtered to return specific search results by appending “/?”and any of the below parameters, such as `/?types=REFRESH`. Multiple search parameters can be joined by the “&” symbol. These filters and the values they support are:

Parameters	Data Type	Description
locationCode	string	The identifier of the location that the inventory job is applying to.
limit	integer	The maximum amount of search results that may be returned.
types	string(s)	The type(s) of job being performed, such as an inventory refresh or adjust, as either a single value or a list. See below for the full list of supported types.
originalFilename	string	The original file name of the uploaded inventory data for the job.

This is the full list of supported job types. The query is case-sensitive, so the type must always be fully capitalized.

- REFRESH
- ADJUST
- DELETE
- ALLOCATE
- DEALLOCATE
- FULFILL
- RELEASE_SHIPMENTS
- CREATE_PICK_WAVE
- CLOSE_PICK_WAVE
- PUT_AWAY_FILE
- SHORT_PICK_ERROR
- BIN_AUDIT_START
- BIN_AUDIT_COMPLETION
- CREATE_BIN_AUDIT
- RELEASE_PENDING_ITEMS

Job Query Example

This example queries for all inventory refresh jobs at an “example” location.

```
https://t{tenantId}.{host}/api/commerce/inventory/v1/queue/?types=REFRESH&locationCode=example
```

After sending the request, the API responds with a collection of job objects as the search results.

You should always use `locationCode` instead of `locationID` when performing queries,

> even if the response returns the ID such as in the below example.


```
[
   {
      "jobID": 00000,
      "tenantID": 11111,
      "locationID": 01010,
      "type": "REFRESH",
      "added": "2020-03-13T21:01:48+0000",
      "started": "2020-03-13T21:01:48+0000",
      "finished": "2020-03-13T21:01:48+0000",
      "hasData": true,
      "itemCount": "1",
      "status": "SUCCESS",
      "success": true
   },
   {
      "jobID": 11111,
      "tenantID": 11111,
      "locationID": 01010,
      "type": "REFRESH",
      "added": "2020-03-13T19:59:58+0000",
      "started": "2020-03-13T19:59:58+0000",
      "finished": "2020-03-13T19:59:58+0000",
      "hasData": true,
      "itemCount": "1",
      "status": "SUCCESS",
      "success": true
   }
]
```


Granular Fields in the API

When performing the below API actions, use these fields to manage granular records if you are using those in your inventory configuration.

- Allocate Inventory and Create Order: You can specify `condition` and `serialNumber` fields at the item level.
- Deallocate Inventory: If you want to deallocate from a specific granular inventory record, include an `inventoryID` for each item on the item level. The `inventoryIDs` can be found in the shipment data.
- Fulfill Inventory: If you want to fulfill inventory from a specific granular record, include an `inventoryID` for each item on the item level. The `inventoryIDs` can be found in the shipment data.
- Delete Inventory: If you want to delete a specific granular record, include the field (such as `date` or `lotCode`) in the request. The API model in the linked documentation will be updated with these fields soon.
    - The optional `deleteGranularRecord` flag can help delete records not associated with specific granular values. This flag should only be provided when you are not specifying

any granular fields.
    - If `deleteGranularRecord` is set to true, then only the granular record without any granular values will be deleted. If set to false, then the request is applied at the UPC-Location level which will be deleted along with all of its granular records.
- Update Inventory Allocation (doc coming soon): Inventory allocation information is tracked at the item level of Shipment APIs, but sometimes discrepancies like damaged inventory or mismatched data may require reallocation. The `.../commerce/shipments/{shipmentId}/items/{itemId}/updateInventoryAllocation` API can be used to adjust granular quantity and add new granular records by updating the `inventoryAllocationInfo` array.
    - Note that you cannot change the granular records of any item with a serial number, as these are always a unique item with an inventory of 1 and cannot be replaced.

Get Inventory Options

When calling Get Inventory, use the below flags to fine-tune which fields are returned in the results. The API documentation will be updated soon to add these fields to the model.

- If `showGranularInventory` is not provided, the On Hand inventory calculation will include Safety Stock.
- If `showGranularInventory` is provided, then the response will include granular inventory records but exclude the granular-levels Safety Stock, LTD, and Floor values.
- If `showGranularInventory` and `showExtraGranularProductFields` are both provided, the response will display both granular inventory fields and the granular-level Safety Stock, LTD, and Floor values. However, Safety Stock will not be included in the calculation of On Hand inventory.

```
{
  "type": "ALL",
  "items": [
    {
      "upc": "upc-1",
      "sku": null
    }
  ],
  "pageNum": 1,
  "pageSize": 100,
  "showGranularInventory": true,
  "showExtraGranularProductFields": true
}
```

Segmentation in the API

Segmentation brings more flexibility to inventory management, allowing each inventory record to

be separated into different categories to indicate that portions of its total quantity are intended for different channels, customer groups, fulfillment methods, or other needs. This allows for:

- Setting a percentage of the quantity that would be available for each category.
- Setting discrete units at the location level as available for each category.

Segmentation is determined with tags, which identify the groups that inventory must be split into. For example, tags could define how much of each inventory record is set aside for a certain sales channel: the Kibo storefront, Walmart, or Amazon. The percentages of the inventory allotted for each channel would add up to 100% – the Kibo storefront could have 80% of the inventory, Amazon 10%, and Walmart 10%.

For more information about segmentation and how these configurations are reflected in the user interface, see the general Inventory Management feature guide.

Tag Categories

You should first create a category for your tags. For example, the below would create a category with three tags. Any number of additional tags could be added to the new category, and the percent values (the percentage of the total quantity that should be allotted to this tag value) could be excluded if they do not need to be enforced.

```
{
  "name": "Channel",
  "tenantID": 1,
  "tags": [
    {
      "tagValue": "Kibo",
      "isDefault": true,
      "percent": 50
    },
    {
      "tagValue": "Amazon",
      "isDefault": false,
      "percent": 25
    },
    {
      "tagValue": "Walmart",
      "isDefault": false,
      "percent": 25
    }
}
```

You can then retrieve, edit, and delete the category as necessary with additional POST (with the same request format), GET, and DELETE calls.

Tag Values

Once a tag category exists, tags can also be managed at the individual level rather than updating the entire category every time. However, this is still a call made to the Tag Category API – the name of the category is just appended, followed by the tag resource. For example, the below URL

would add a tag to the Channel category that was just created:

```
../v1/tagCategory/Channel/tag
```

The tag would be defined in the request body (as with the categories, the percent is optional):

```
{
   "tagValue": "Ebay",
   "isDefault": false,
   "percent": 10,
}
```

You can then retrieve, edit, and delete the values as necessary with additional POST (with the same request format), GET, and DELETE calls.

Tags in the Inquiry Response

If tagged inventory is being utilized for inventory segmentation, then the future inventory data will be placed within the `taggedInventory` object of an inquiry. For example, this would be a full request with tagged and future inventory:

```
[
   {
      "locationName": "Example2",
      "locationCode": "00002",
      "tenantID": 11111,
      "onHand": 0,
      "available": 0,
      "allocated": 0,
      "pending": 0,
      "upc": "InventoryTagTest",
      "blockAssignment": false,
      "ltd": 0,
      "floor": 0,
      "safetyStock": 0,
      "distance": 0,
      "directShip": true,
      "transferEnabled": false,
      "pickup": true,
      "countryCode": "US",
      "currencyID": 1,
      "retailPrice": 1.99,
      "taggedInventory": [
         {
            "onHand": 0,
            "available": 0,
            "allocated": 0,
            "pending": 0,
            "tags": {
               "HolidayItems": "HolidayTest1"
            },
            "futureInventory": [
               {
                  "futureInventoryID": 3,
                  "quantity": 150,
                  "type": "Refresh",
                  "deliveryDate": "2021-03-20T17:00:00+0000",
                  "createDate": "2021-03-08T11:52:32+0000"
               }
            ]
         }
      ]
   }
]
```


Tags in the Create Order Request

Multiple tag names and values can be passed in the Create Order call as part of order item data. This will allow the inventory, order routing, and fulfillment services to allocate inventory appropriately. The values included in the `inventoryTags` object will be validated against existing tags in the system, and if no tags are passed then the order item will be set to the default tag.


```
{
 "isTaxExempt": false,
 "email": "example@kibo.com",
 "ipAddress": "172.16.254.167",
 "type": "Offline",
 "externalId": "tags",
 "isEligibleForReturns": false
```

```
 "isEligibleForReturns": false,
"totalCollected": 0,
"attributes": [],
"shippingDiscounts": [],
"handlingDiscounts": [],
"handlingTotal": 0,
"notes": [],
"items": [
 {
  "id": "eb75a8fa9a354e66ae32ada300c9c5a1",
  "fulfillmentLocationCode": "Loc1",
  "fulfillmentMethod": "Ship",
  "lineId": 1,
  "product": {
   "fulfillmentTypesSupported": [
    "DirectShip",
    "InStorePickup"
   ],
   "options": [],
   "properties": [],
   "categories": [],
   "price": {
    "price": 10
   },
   "discountsRestricted": false,
   "isTaxable": true,
   "productType": "AllUsage",
   "productUsage": "Standard",
   "bundledProducts": [],
   "productCode": "sp_01",
   "name": "sp_01",
   "goodsType": "Physical",
   "isPackagedStandAlone": false,
   "stock": {
    "manageStock": true,
    "isOnBackOrder": false,
    "stockAvailable": 100,
    "aggregateInventory": 0
   },
   "measurements": {
    "height": {
     "unit": "in",
     "value": 1
    },
    "width": {
     "unit": "in",
     "value": 1
    },
    "length": {
     "unit": "in",
     "value": 1
    },
    "weight": {
     "unit": "lbs",
     "value": 1
    }
   },
   "fulfillmentStatus": "PendingFulfillment"
  },
  "quantity": 1,
```

```
  "subtotal": 10,
  "extendedTotal": 10,
  "taxableTotal": 10,
  "discountTotal": 0,
  "discountedTotal": 10,
  "itemTaxTotal": 0,
  "shippingTaxTotal": 0,
  "shippingTotal": 0,
  "feeTotal": 0,
  "total": 10,
  "unitPrice": {
   "extendedAmount": 10,
   "listAmount": 10
  },
  "productDiscounts": [],
  "shippingDiscounts": [],
    "shippingAmountBeforeDiscountsAndAdjustments": 0,
  "weightedOrderAdjustment": 0,
  "weightedOrderDiscount": 0,
  "adjustedLineItemSubtotal": 10,
  "totalWithoutWeightedShippingAndHandling": 10,
  "weightedOrderTax": 0,
  "weightedOrderShipping": 15,
  "weightedOrderShippingDiscount": 0,
  "weightedOrderShippingManualAdjustment": 0,
  "weightedOrderShippingTax": 0,
  "weightedOrderHandlingFee": 0,
  "weightedOrderHandlingFeeTax": 0,
  "weightedOrderHandlingFeeDiscount": 0,
  "weightedOrderDuty": 0,
  "totalWithWeightedShippingAndHandling": 10,
  "weightedOrderHandlingAdjustment": 0,
  "isAssemblyRequired": false,
  "inventoryTags": [{
   "name": "Online",
   "value": "Flipkart"
  },
  {
   "name": "Offline",
   "value": "OnlinePurchase"
  }
 ]
 }
],
"validationResults": [],
"billingInfo": {
 "billingContact": {
  "id": 1000,
  "email": "example@kibo.com",
  "firstName": "anagha",
  "middleNameOrInitial": "",
  "lastNameOrSurname": "deshmukh",
  "companyOrOrganization": "Test",
  "phoneNumbers": {
   "home": "5129991111",
   "mobile": "",
   "work": ""
  },
  "address": {
   "address1": "717 N. Harwood St.",
```

```
  "address1": "717 N. Harwood St.",
  "address2": "",
  "address3": "",
  "address4": "",
  "cityOrTown": "Dallas",
  "stateOrProvince": "TX",
  "postalOrZipCode": "75201",
  "countryCode": "US",
  "addressType": "Residential",
  "isValidated": false
 }
},
"isSameBillingShippingAddress": false

},
"payments": [
{
      "paymentType": "CreditCard",
      "paymentWorkflow": "Mozu",
      "billingInfo": {
        "paymentType": "CreditCard",
        "billingContact": {
          "email": "sam.billing@email.com",
          "firstName": "Sam",
          "middleNameOrInitial": "",
          "lastNameOrSurname": "Billing",
          "phoneNumbers": {
            "home": "1234567895",
            "mobile": "1234567895",
            "work": ""
          },
          "address": {
            "address1": "1845 Kramer Ln",
            "address2": "",
            "address3": "",
            "address4": "",
            "cityOrTown": "Austin",
            "stateOrProvince": "TX",
            "postalOrZipCode": "78758",
            "countryCode": "US",
            "addressType": "Residential",
            "isValidated": false
          }
        },
        "isSameBillingShippingAddress": false,
        "card": {
          "isUsedRecurring": false,
          "nameOnCard": "Sam Billing",
          "isCardInfoSaved": false,
          "paymentOrCardType": "VISA",
          "cardNumberPartOrMask": "4111111111111111",
          "isTokenized": true,
          "expireMonth": 1,
          "expireYear": 2023
        },
        "auditInfo": {
          "updateDate": "2018-09-17T17:54:21.027Z",
          "createDate": "2018-09-17T17:53:24.598Z",
          "updateBy": "355060a60a5e48eeb7f2fb8d92af2ba5",
          "createBy": "355060a60a5e48eeb7f2fb8d92af2ba5"
```

```
        }
      },
      "status": "Authorized",
      "subPayments": [],
      "interactions": [
        {
          "currencyCode": "USD",
          "interactionType": "Authorization",
          "status": "Authorized",
          "paymentEntryStatus": "New",
          "isRecurring": false,
          "isManual": false,
          "gatewayTransactionId": "40018563192",
          "gatewayAuthCode": "ASQLVP",
          "gatewayAVSCodes": "Y",
          "gatewayCVV2Codes": "P",
          "gatewayResponseCode": "1",
          "gatewayResponseText": "This transaction has been approved.",
          "gatewayResponseData": [
            {
              "key": "AuthorizationRequestId",
              "value": "5642607567736184003013"
            },
            {
              "key": "AuthorizationRequestToken",
              "value": "Ahj/7wSTMeLpFwo2JFnENyzVs0ZNmDdq2bt2bZi4aMGDNgxZKNP3ock7AVGn
70OSd6QNnBx/EMmkmW6QHez7QJyZjxdIuFGxIs4gLxmU"
            },
            {
              "key": "currencyCode",
              "value": "USD"
            }
          ],
          "amount": 65.1,
          "interactionDate": "2018-09-17T17:54:21.098Z"
        }
      ],
      "isRecurring": false,
      "amountCollected": 0,
      "amountCredited": 0,
      "amountRequested": 65.1
    }
],
"refunds": [],
"credits": [],
"packages": [],
"pickups": [],
"digitalPackages": [],
"isDraft": false,
"hasDraft": false,
"isImport": true,
"isHistoricalImport": false,
"isUnified": true,
"couponCodes": [],
"invalidCoupons": [],
"amountAvailableForRefund": 0,
"amountRemainingForPayment": 0,
"amountRefunded": 0,
"readyToCapture": false,
"isOptInForSms": false,
```

```
 "isOptInForSms": false,
"userId": "82bb750a90d34ed1b1ddc2c534293773",
"id": "1240f01063e4ce00014e0aa7000047f0",
"tenantId": 18416,
"siteId": 23245,
"channelCode": "Online",
"currencyCode": "USD",
"customerInteractionType": "Unknown",
"fulfillmentInfo": {
 "fulfillmentContact": {
  "id": 1000,
  "email": "example@kibo.com",
  "firstName": "anagha",
  "middleNameOrInitial": "",
  "lastNameOrSurname": "deshmukh",
  "companyOrOrganization": "Test",
  "phoneNumbers": {
   "home": "5129991111",
   "mobile": "",
   "work": ""
  },
  "address": {
   "address1": "717 N. Harwood St.",
   "address2": "",
   "address3": "",
   "address4": "",
   "cityOrTown": "Dallas",
   "stateOrProvince": "TX",
   "postalOrZipCode": "75201",
   "countryCode": "US",
   "addressType": "Residential",
   "isValidated": false
  }
 },
 "shippingMethodCode": "d0fb5d2e4fc047d596aaada300c05893",
 "shippingMethodName": "Flat Rate",
 "auditInfo": {
  "updateDate": "2021-09-14T12:14:46.532Z",
  "createDate": "2021-09-14T12:14:08.129Z",
  "updateBy": "355060a60a5e48eeb7f2fb8d92af2ba5",
  "createBy": "355060a60a5e48eeb7f2fb8d92af2ba5"
 }
},
"orderDiscounts": [],
"suggestedDiscounts": [],
"subtotal": 10,
"discountedSubtotal": 10,
"discountTotal": 0,
"discountedTotal": 10,
"shippingTotal": 15,
"shippingSubTotal": 15,
"shippingTaxTotal": 0,
"handlingTaxTotal": 0,
"itemTaxTotal": 0,
"taxTotal": 0,
"feeTotal": 0,
"total": 10,
"lineItemSubtotalWithOrderAdjustments": 10,
"shippingAmountBeforeDiscountsAndAdjustments": 15,
"lastValidationDate": "2021-09-14T12:14:42.613Z",
```

```
 "extendedProperties": [],
 "discountThresholdMessages": [],
 "auditInfo": {
  "updateDate": "2021-09-14T12:15:03.682Z",
  "createDate": "2021-09-14T12:14:08.162Z",
  "updateBy": "355060a60a5e48eeb7f2fb8d92af2ba5",
  "createBy": "355060a60a5e48eeb7f2fb8d92af2ba5"
 }
}
```

Inventory Redistribution

The Inventory Redistribution API allows on-demand redistribution of inventory levels across segmentation tags based on the allocation percentages that you defined for them. This lets you react quickly to inventory depletion in sales channels, as this API will trigger asynchronous redistribution at the next Refresh or Adjust update and restore the appropriate amount of inventory to those channels again.

To do this, make a POST call to `.../v1/inventory/redistribute` with a list of UPCs and/or location codes as shown in the below example. The API documentation will be updated with this endpoint soon. The scope of the redistribution depends on the combination of information you provide:

- If you provide only UPCs, then those UPCs across all locations will be distributed.
- If you provide only locations, then all UPCs at those locations will be redistributed.
- If you provide both UPCs and locations, then only those UPCs at those locations will be redistributed.
- If you provide an empty request, then all UPCs across all locations on your tenant will be redistributed.

```
{
 "upcs": ["UPC1", "UPC2", ...],
 "locationCodes": ["Location1", "Location2", ...]
}
```

A successful response will return a Job Id, allowing you to check the status via the Get Job API.

If a redistribution for a UPC/Location would cause any source tag's available quantity to decrease below zero, then the redistribution for that entire UPC/Location combination will be skipped.