Eusis WMS Integration with Pharmacy Software

Index

[Index 2](#_Toc527041391)

[Architecture 5](#_Toc527041392)

[Integrations scenarios 6](#_Toc527041393)

[Registries 6](#_Toc527041394)

[Loading of a whole supplier bill/Return from ward 7](#_Toc527041395)

[WMS provided refill 7](#_Toc527041396)

[Customer IT Systems provided refill 8](#_Toc527041397)

[Direct returns 8](#_Toc527041398)

[Ward replenishment unload with manual choice of the exit 9](#_Toc527041399)

[Ward replenishment unload with automatic choice of the exit 9](#_Toc527041400)

[Direct unload 10](#_Toc527041401)

[Overtime/direct unloading 10](#_Toc527041402)

[Customer IT Systems required Inventory/Stock Recheck 11](#_Toc527041403)

[WMS provided Inventory/Stock Recheck 11](#_Toc527041404)

[Custom Middleware 12](#_Toc527041405)

[Loading of whole supplier bill: bill 12](#_Toc527041406)

[Item registry 12](#_Toc527041407)

[Loading of whole supplier bill: order 12](#_Toc527041408)

[Unload request 12](#_Toc527041409)

[Unload movements after a request 13](#_Toc527041410)

[Inventory 13](#_Toc527041411)

[Conventions 13](#_Toc527041412)

[Security 13](#_Toc527041413)

[Exposed by 13](#_Toc527041414)

[URL 13](#_Toc527041415)

[Method 14](#_Toc527041416)

[Input: 14](#_Toc527041417)

[Output: 14](#_Toc527041418)

[Items 15](#_Toc527041419)

[Exposed by 15](#_Toc527041420)

[URL 15](#_Toc527041421)

[Method 15](#_Toc527041422)

[Input: 15](#_Toc527041423)

[Output: 18](#_Toc527041424)

[Pickup Centers 19](#_Toc527041425)

[Exposed by 19](#_Toc527041426)

[URL 19](#_Toc527041427)

[Method 19](#_Toc527041428)

[Input: 19](#_Toc527041429)

[Output: 21](#_Toc527041430)

[Causes 21](#_Toc527041431)

[Exposed by 21](#_Toc527041432)

[URL 22](#_Toc527041433)

[Method 22](#_Toc527041434)

[Input: 22](#_Toc527041435)

[Output: 23](#_Toc527041436)

[Refill List 23](#_Toc527041437)

[Exposed by 24](#_Toc527041438)

[URL 24](#_Toc527041439)

[Method 24](#_Toc527041440)

[Input: 24](#_Toc527041441)

[Output: 25](#_Toc527041442)

[Warehouse Stock 25](#_Toc527041443)

[Exposed by 26](#_Toc527041444)

[URL 26](#_Toc527041445)

[Method 26](#_Toc527041446)

[Input: 26](#_Toc527041447)

[Output: 26](#_Toc527041448)

[Robot Stock 27](#_Toc527041449)

[Exposed by 27](#_Toc527041450)

[URL 27](#_Toc527041451)

[Method 27](#_Toc527041452)

[Input: 27](#_Toc527041453)

[Output: 27](#_Toc527041454)

[Unload Requests 28](#_Toc527041455)

[Exposed by 28](#_Toc527041456)

[URL 28](#_Toc527041457)

[Method 28](#_Toc527041458)

[Input: 28](#_Toc527041459)

[Output: 30](#_Toc527041460)

[Direct Unload Request 31](#_Toc527041461)

[Exposed by 31](#_Toc527041462)

[URL 31](#_Toc527041463)

[Method 31](#_Toc527041464)

[Input: 31](#_Toc527041465)

[Output: 33](#_Toc527041466)

[Movements 34](#_Toc527041467)

[Exposed by 34](#_Toc527041468)

[URL 34](#_Toc527041469)

[Method 34](#_Toc527041470)

[Input: 34](#_Toc527041471)

[Inventory Request 37](#_Toc527041472)

[Exposed by 37](#_Toc527041473)

[URL 37](#_Toc527041474)

[Method 37](#_Toc527041475)

[Input: 37](#_Toc527041476)

[Output: 37](#_Toc527041477)

[Real Stock 38](#_Toc527041478)

[Exposed by 38](#_Toc527041479)

[URL 38](#_Toc527041480)

[Method 38](#_Toc527041481)

[Input: 38](#_Toc527041482)

[Output: 39](#_Toc527041483)

## Architecture

The integration layer of the architecture provides the maximum flexibility while allowing standard and stable integration patterns.

Customer IT Systems



Custom Middleware

Core Middleware

DB

It’s possible to interact with the system in 2 different ways:

* The Customer IT Systems interact directly with the standard services offered by the Core Middleware
* The Customer IT Systems interacts with a custom middleware, built to minimize the development on the Customer IT Systems that translates the different calls to pre existing interfaces on the Customer IT Systems. If data needs to be enriched (i.e. the core middleware sends only information about the code of an item but also the name of the item is needed), the custom middleware can access the DB to retrieve them.

Since the Custom Middleware is specific for the installation of the customer and will be written evaluating the specific needs of the customer and the specific technology used by the customer (Rest Services, Soap Services, HL7, SAP BAPI etc).

Normally a test environment is required in the environment of the customer and the possibility to migrate data from the live system to the test system.

A new version of the WMS (including the integration) will be deployed only after a successful test session in this environment.

The Core Middleware interacts directly with the Database and hardware and provides a set of standard REST services. The custom middleware translates and manages the calls (incoming and outcoming) from the core middleware to the calls specific for the Custom IT System.

In this chapter the full set of the interfaces is described in technical details, giving the exact specification to each field and examples to each call.

## Integrations scenarios

In this chapter a quick overview of the integration scenarios is outlined, in particular with a use case whose integration flows are involved and in which way.

Other scenarios are possible given the current integration flows, but the most common ones are described here.

In the following diagrams the name over the arrow is the name of the integration flow listed below.

For clarity when the robot or the user are involved they are present in the diagrams,

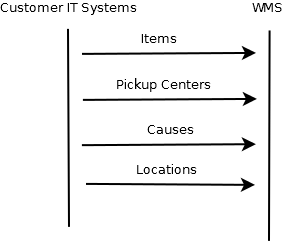
### Registries

At Go Live time the full registry must be supplied to the WMS software and then regularly when something changes.

The master of all this data is the Customer IT Systems.

The registries to align are:

* Items (including barcodes, storage condition, default warehouse location, manageable in the robot etc)
* Pickup centers (i.e. destinations for the goods unloaded etc)
* Causes (the causes both for loads and for unloads). This integration is optional and can be managed as a configuration since causes changes very seldomly
* Warehouse locations. This integration is optional and can be managed as a configuration since physical locations changes seldomly)



The information is then saved on the middleware and used for later integrations (for example to know what the barcodes of the items are et.)

In case some important information is not available in the Customer IT Systems (for example if the item is manageable in the robot ) the information can be augmented on the WMS software.

### Loading of a whole supplier bill/Return from ward

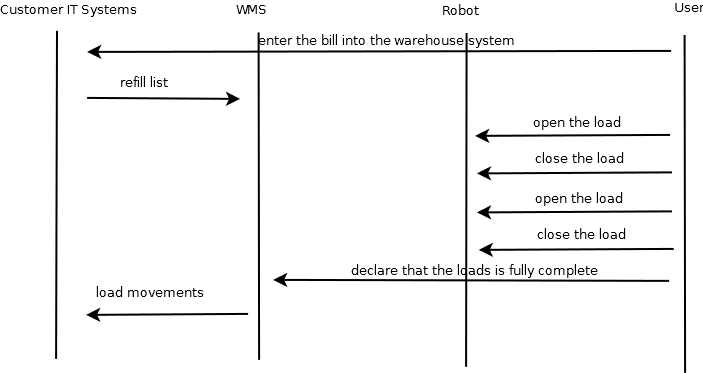
In this scenario the user (or some automatic system) load into the Customer IT Systems a load document. This document can be for example a supplier bill, a return request from a ward etc etc

The user then sends the load document to the WMS software.

Here it’s used to load the packs into the robot. The user can load, interrupt to load something else, load as many times as he wishes.

In the end, when the user declares that all the material is loaded, the WMS is going to send back to the customer IT System.

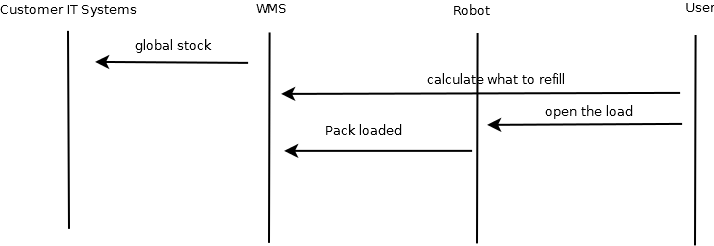
NB: this scenario is really useful only if the documet contains a not so big number o packs, since loading a big number of packs of the same reference takes a lot of time and it’s not very useful and uses space in the robot that can be better used for other things.



### WMS provided refill

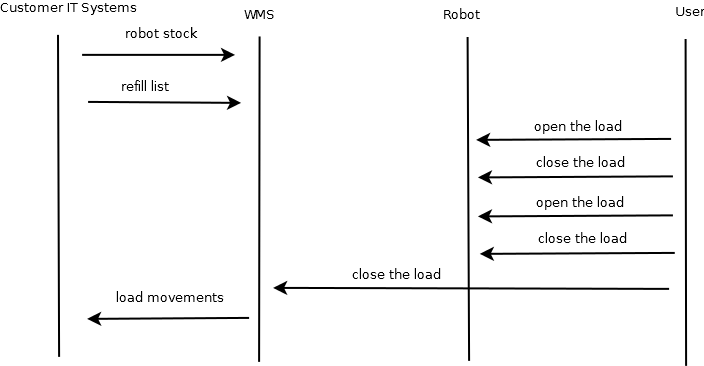
In this scenario the WMS software asks to the Customer IT Systems the stock of the whole warehouse, then by difference it calculates the stock outside the robot and then it creates a loading list for the Robot (based on past usage and available stock), or the user creates the loading list manually (aided by the knowledge of the outside stock).

Then the user loads the packs (or uses the AIS to load them).



### Customer IT Systems provided refill

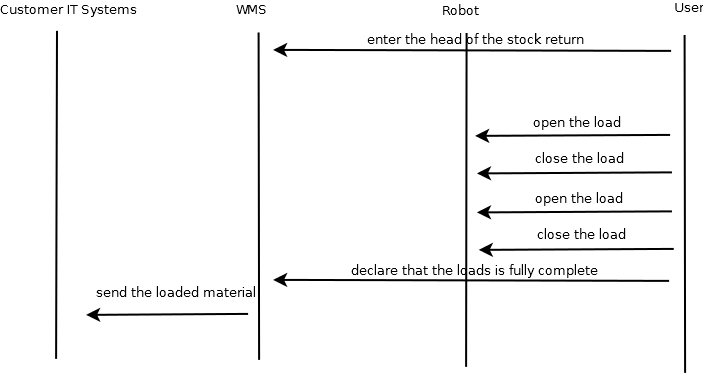
In this scenario the Customer IT System asks for the stock into the robot, and then provides a loading list to the WMS software. The WMS software, based on historical consumption and manual thresholds calculate for each item an optimal stock in the robot and an alarm level for the stock, when the user should load more packs for a given item. This information, together with the external stock of the warehouse, is the input used by the WMS software to calculate a loading list for the robot. The user then loads the packs (or uses the AIS to load them).



### Direct returns

In this scenario the user creates in the WMS software the head of a return from a ward (typically the pickup center that returns the goods, and the cause of the returns).

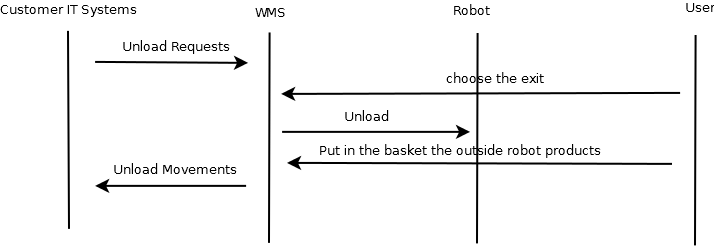
Than it goes to the robot and load the material. In the end the document is closed by the user on the WMS software that sends the return to the Customer IT Systems where it’s registered.



### Ward replenishment unload with manual choice of the exit

For ward replenishment the unloads are not immediately processed, are big and must not be mixed together. The Customer IT Systems sends the unload requests. The user chooses when to process the unloads, choosing an exit from the WMS software. The robot is required to unload the stock (eventually changing the basket if needed), and the user fill in the missing products. The user adds the missing products.

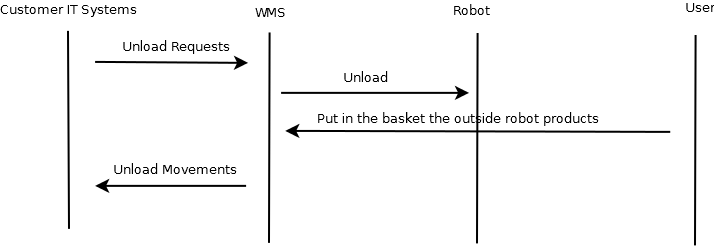
When the unload is finished the whole set of movements is sent to the Customer IT Systems.



### Ward replenishment unload with automatic choice of the exit

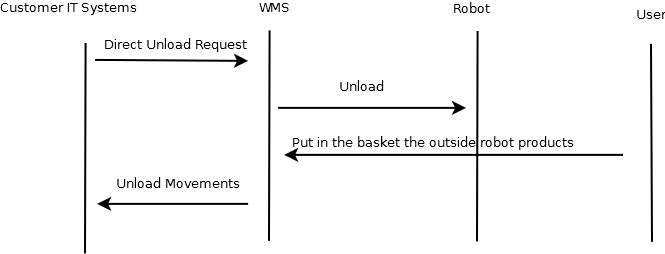
For ward replenishment the unloads are not immediately processed, are big and must not be mixed together. The Customer IT Systems sends the unload requests. When the robot is not unloading and the unload is the first in the FIFO list, the WMS asks to the robot to unload the packs (eventually managing the fact that it’s split in multiple baskets). The user adds the missing products.

When the unload is finished the whole set of movements is sent to the Customer IT Systems



### Direct unload

Direct unloads are normally controlled by a user. They must start immediately without any delay. The Customer IT Systems sends to the WMS what to unload, the unload starts and, when the unload is finished, the whole set of movements is sent to the Customer IT Systems.



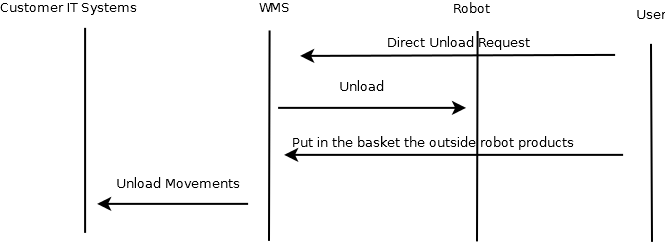
### Overtime/direct unloading

The unload can happen directly from the WMS systems in 2 cases:

1. It’s communicated to the warehouse using out of procedure methods (fax, phone)
2. Using BancoPharma for nightly /weekend unloads.

In this case a user creates an unloads in the WMS software (choosing items, quantity, pickup center and cause), which asks to unload to the robot.

When the unload is finished the whole set of movements is sent to the Customer IT Systems.

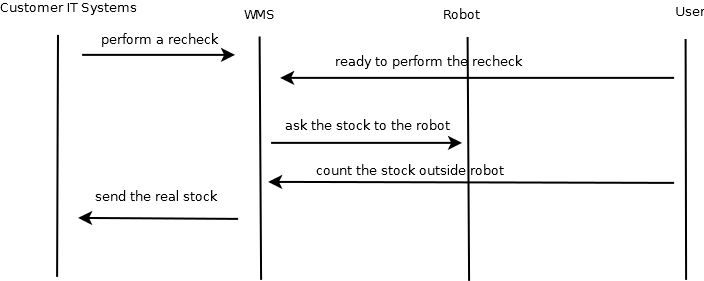


### Customer IT Systems required Inventory/Stock Recheck

The customer IT Systems can ask for an inventory of the whole stock or only of some items. The request is queued until a user can process it. The WMS software will ask to the Customer IT Systems the stock of the whole warehouse, will ask to the Robot the stock of the Robot and will ask to the user to check the stock of the items outside the robot.

The WMS will send back to the Customer IT Systems the stock.

This scenario is usually used when a whole inventory (useful also for administrative reason) is required.

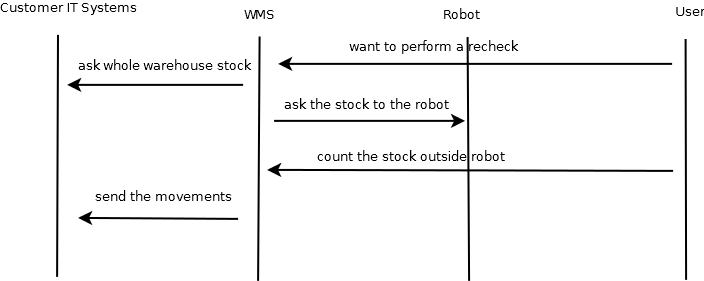


### WMS provided Inventory/Stock Recheck

In the WMS software a recheck is performed by the user(for example to check for a stock discrepancy).The WMS software ask the stock of the whole warehouse to the Customer IT systems, then ask the stock to the robots and to the user check the outside stock the robot.

In the end WMS software sends back the movements to the Customer IT Systems.

This scenario is mostly used when a stock discrepancy is corrected on the WMS system.



## Conventions

Every call will return an HTTP status code of 200 if it succeeds, other status codes otherwise.

In particular:

500: an error (managed or not managed) was encountered

401: authorization needed. The authorization token (see following chapter) is missing, wrong or expired.

If the Content-Type http response header is text/plain the response body is a simple text that should not be parsed (typically an error message), if the Content-Type http response header is application/json the response body is a JSON that should be parsed.

The format of the date used in the JSON fields is YYYY-MM-DD**T**HH:mm:ss

For example, September 1st, 2018, at 21:14 and 32 seconds will be

2018-09-01T21:14:32

## Security

There are 3 different aspects of security considered in this interface: integrity, confidentiality and authorization.

Integrity and confidentiality are ensured using the standard HTTPS protocol.

Authorization is ensured by introducing a login service that returns an authorization token to use in the following calls.

The authorization token must be sent in all subsequent calls using a custom HTTP header Authorization that must contain the following string:

Bearer: <authorization token>

### Exposed by

Core Middleware

### URL

/middleware/login

### Method

POST

### Input:

The following http parameters are sent to the service:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| login | string | Login used to authenticate (validate) | Yes |
| password | string | Password used to authenticate (validate) | Yes |

### Output:

HTTP Status Codes:

200 = Success

401= Login failed

398 = Password expired

If the http status 200 is returned the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| accessToken | string | The token to use in the following requests | Yes |
| expiresIn | Integer | The number of minutes this token will be valid from the moment it was generated | Yes |

Example:

{

"success": true,

"accessToken": "eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ZuS7FibrOueqqW4AOt-H8yVfb8b-SGBl7Q0slwO7xPxI8KKz9dI-3g.GWw-Egd0lpIFckvrJsJ6kQ.vS5-lhb3XyZzKT1lgqHmd5TPE-Jmj-Hl7FcngcqLE8QgOhQcQfP5ahVmElKq49z-q7-EIlZ4JMcxtnyQRxsubJxrk\_gujy3Oj6EVuCQJHDGxbamK-oFusYpX9rSpAFxI1Oh0Z6KcWTgRkbj7tDQUlN8DoPBxo4ZKaOl7ZqLXcNqFuWrS\_7qMuqRR\_2cts6Mdo8P1Qc7ExyTScDrlxMb5h928JgPIe-6QMQf6zI-ivMHKvvbSLB4SeefV7bSMpL2jnwZw7\_f8K0kH7S4UXk7B-A.UMskOE2fjBGowXQ2a9RyKw",

"expiresIn: "10"

}

## Items

This service takes as input an item list and inserts/updates/disables the items. It is exposed by the Core middleware to allow the maximum implementation flexibility from the Custom Middleware (for example the Custom Middleware can be called from the Customer IT system when a change is performed to an item, in this case the Custom Middleware will simply forward the call, or the Custom Middleware can periodically request the Customer IT system for changed items and then call the service on the Core Middleware with these items.

The items are managed transactionally one by one (which means that a failure on a single item causes a full rollback for that item, the others will not be affected).

If a non-mandatory field is missing during insertion, it will be valued with the default value, during modification the field will not be changed.

### Exposed by

Core Middleware

### URL

/middleware/items

### Method

POST

### Input:

A JSON with the following structure is expected:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** | Default value |
| items | Object Array | List of items | Yes |  |
| code | string | Code of the item (Item Code). This code identifies the item in all the following services, and cannot change between the calls | Yes |  |
| description | String | Description of the item (Item Description). Only the first 40 characters will be shown on the robot interface | Yes |  |
| latinDescription | String | Description of the items in Latin characters. To be used if the description is made in “non-latin” characters (Cyrillic, Arabic etc) | No |  |
| fragile | Boolean | True if the item is fragile, false otherwise. Used when unloading from the robots in baskets to allow that no heavy item falls on a fragile one. | No | false |
| heavy | Boolean | True if the item is heavy, false otherwise. Used when unloading from the robots in baskets to allow that no heavy item falls on a fragile one. | No | false |
| stockingType | Integer | 0: Normal, 1: Fridge Other stock types can be added as needed. | No | 0 |
| allowedInRobot | Boolean | True: the item is allowed in the robot, False: the item is not allowed in the robot. |  | true |
| blocked | Boolean | True: the item is blocked and cannot be loaded/unloaded | No | false |
| managedByBatch | Boolean | True: for this item the stock must be maintained by batch number and expiry date  False: for this item the stock should not be maintained by batch number and expiry date | No | false |
| um | Object | Unit of measure. This is the unit of measure that will be used in all the following services to convert to and from pack | Yes |  |
| code | String | Code of the unit of measure. Max length 6 characters (for example PACK, TBL for tablets, CPS for CAPSULE, VIAL for vial and so on) | Yes |  |
| description | String | Description of the unit of measure | No |  |
| packQuantity | Integer | Number of measure unit in a pack. For a virtual item (se virtual field) always 1 | Yes |  |
| virtual | Boolean | If true, this item is a template product for other commercial products. For example, ACETILSALICILIC ACID 40 mg tablets is a template for BAYER ASPIRIN 40mg tablets. Virtual items can be used in the unload requests to let the system choose which commercial item will be unloaded | No | false |
| commercialItemsCodes | String Array | The list of the commercial items equivalent to this item. If this item is not a virtual it must be null | No |  |
| barcodes | Object Array | List of the barcodes for this item | No (but an item without barcode cannot be loaded into the robot) |  |
| barcode | string | The barcode of the item | Yes |  |
| isDefault | Boolean | The barcode that will be used by the robot to identify this item. An item can have more than one barcode, but after the loading in the robot the item will be identified by this barcode. This barcode must never be changed if there is stock in the robot. Trying to do so will produce an error in the response of the service. An item must have exactly a default barcode | Yes |  |
| barcodeType | String | The type of the barcode. For example EAN, GTIN etc | No |  |
| type | Object | type of the items (DRUG, MEDICAL DEVICE etc.) | No |  |
| typeCode | string | Type Code. Maximum length 6 characters | Yes |  |
| typeDescription | string | Type description | Yes |  |
| serialNumberMandatory | Boolean | Loading into the robots is mandatory to read the serial number. Used only when item barcode and serial number are in different barcodes (multiline GS1-128 etc) | No | false |
| overtimeDispensable | Boolean | If true and the Overtime dispensing module is enabled, the product will be enabled for overtime dispensing, otherwise the module will not be enable for overtime dispensing | No | false |

Example:

{

"items": [{

"code": "AB12",

"description": "ACETILSALICILIC ACID 500 MG TABLETS",

"latinDescription": null,

"fragile": null,

"heavy": null,

"stockingType": null,

"allowedInRobot": false,

"blocked": false,

"managedByBatch": null,

"um": {

"code": "TBL",

"description": "TABLET"

},

"packQuantity": 1,

"virtual": true,

"commercialItemsCodes": ["AB12-1", "AB12-2"],

"barcodes": null,

"type": {

"typeCode": "DRUG",

"typeDecription": "Drug"

},

"serialNumberMandatory": null,

"overtimeDispensable": null

}, {

"code": "AB12-1",

"description": "ASPIRIN 500 MG TABLETS",

"latinDescription": null,

"fragile": false,

"heavy": false,

"stockingType": 0,

"allowedInRobot": true,

"blocked": false,

"managedByBatch": true,

"um": {

"code": "TBL",

"description": "TABLET"

},

"packQuantity": 20,

"virtual": false,

"commercialItemsCodes": null,

"barcodes": [{

"barcode": "004763037",

"isDefault": true,

"barcodeType": null

}, {

"barcode": "312843555075",

"isDefault": false,

"barcodeType": null

}

],

"type": {

"typeCode": "DRUG",

"typeDecription": "Drug"

},

"serialNumberMandatory": true,

"overtimeDispensable": true

}

]

}

### Output:

HTTP Status Codes:

200 = Success

If the http status 200 is returned the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| itemsResponse | Object Array | The response of the operation for each item | Yes |
| code | string | code of the item | Yes |
| success | boolean | true: the insertion/change succeeded, false the insertion/change failed | Yes |
| errorMessage | string | The description of the error if success is false | No |

Example:

{

"itemResponse": [{

"code": "AB12",

"success": "true",

"errorMessage": null

},

{

"code": "AB12-1"

"success": "false",

"errorMessage": "Changing the number of units per packs while there is stock in the robot is not allowed"

}]

}

## Pickup Centers

This service takes as input a list of pickup centers and insert/update/disable them.

The pickup centers are managed transactionally one by one (which means that a failure on a single pickup center causes a full rollback for that pickup center, the others will not be affected).

If a non-mandatory field is missing during insertion it will be valued with the default value, during modification the field will not be changed.

Multiple possible implementation scenarios are possible for the custom middleware, for example:

1. Forward the calls from the Customer IT systems
2. When receiving a request to unload an item, call the method to insert/modify the pickup center before calling the method to insert the unload request
3. Define at the beginning only one pickup center and forbid spontaneous unloads from Busterweb Logistics

### Exposed by

Core Middleware

### URL

/middleware/pickupCenters

### Method

POST

### Input:

A JSON with the following structure is expected:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** | Default value |
| pickupCenters | Object Array | A list of pickupCenters | Yes |  |
| code | string | Code of the pickup center. | Yes |  |
| description | String | Description of the pickup center | Yes |  |
| address | String | Default shipment address | No |  |
| groups | Objects | A list of groups to which the user belongs. Used mainly to aid the search of the pickup center in the GUI | No |  |
| groupCode | string | Group code. | Yes |  |
| groupDescription | string | Group description | Yes |  |
| blocked | Boolean | True: the pickup center is blocked and cannot be used | No | false |

Example:

{

"pickupCenters": [{

"code": "1VFCAR",

"description": "CARDIOLOGY",

"address": "Pavilion C, First Floor",

"groups": [{

"groupCode": "1",

"groupDescription": "Monday"

}, {

"groupCode": "4",

"groupDescription": "Thursday"

}

],

"blocked": null

}, {

"code": "1VFMED",

"description": "INTERNAL MEDICINE",

"groups": [{

"groupCode": "2",

"groupDescription": "Tuesday"

}, {

"groupCode": "4",

"groupDescription": "Thursday"

}

],

"blocked": null

}

]}

### Output:

HTTP Status Codes:

200 = Success

If the http status 200 is returned the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| pickupCentersResponse | Object Array | The response of the operation for each pickup center | Yes |
| code | string | code of the pickup center | Yes |
| success | boolean | true: the insertion/change succeeded, false the insertion/change failed | Yes |
| errorMessage | string | The description of the error if success is false | No |

Example:

{

"pickupCentersResponse": [{

"code": "1VFCAR",

"succeas": "true",

"errorMessage": null

}, {

"code": "1VFMED",

"succeas": "false",

"errorMessage": "Error description"

}

]

}

## Causes

This service takes as input a list of movement cause and insert/update/disable them.

The causes are managed transactionally one by one (which means that a failure on a single pickup center causes a full rollback for that pickup center, the others will not be affected).

If a non-mandatory field is missing during insertion it will be valued with the default value, during modification the field will not be changed.

Multiple possible implementation scenarios are possible for the custom middleware, for example:

1. Just forward the calls from the Customer IT systems
2. When receiving a request to unload something/refill the robot call the method to insert/modify the cause before calling the method to insert the unload request
3. Configure on the WMS the mapping between the causes used by the Customer IT system and the ones used internally and that do not use the service

### Exposed by

Core Middleware

### URL

/middleware/causes

### Method

POST

### Input:

A JSON with the following structure is expected:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** | Default value |
| causes | Object Array | A list of causes | Yes |  |
| code | string | Code of the cause. Max length 5 characters | Yes |  |
| description | String | Description of the cause | Yes |  |
| type | string | UNLOAD from the unload causes, REFILL for the refill causes, ORDER for supplier orders, RETURNS for Returns from WARD, RECHECK for inventory/recheck causes . If the system tries to change the type of an already used cause an error is returned.  NB: a return to supplier is an UNLOAD from the WMS software | no | UNLOAD |

Example:

{

"causes": [{

"code": "SCEVR",

"description": "UNLOAD FOR WARD REPLENISHMENT",

"type": "UNLOAD"

}, {

"code": "SCERM",

"description": "SPONTANEOUS UNLOAD",

"type": "UNLOAD"

}, {

"code": "TRREF",

"description": "REFILL TRANSFER",

"type": "UNLOAD"

}

]

}

### Output:

HTTP Status Codes:

200 = Success

If the http status 200 is returned, the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| causesResponse | Object Array | The response of the operation for each cause | Yes |
| code | string | Cause code | Yes |
| success | boolean | True: the insertion/change succeeded; False: the insertion/change failed | Yes |
| errorMessage | string | The description of the error if success is false | No |

Example:

{

"pickupCentersResponse": [{

"code": "SCEVR",

"succeas": "true",

"errorMessage": null

}, {

"code": "SCERM",

"succeas": "false",

"errorMessage": "Error description"

}, {

"code": "TRREF",

"succeas": "false",

"errorMessage": "Error description"

}

]

}

## Refill List

It can be used to send:

* Requests to load external stock into the robot
* An order to supplier for which the good has just arrived and needs to be loaded into the robot
* A return request from a ward that has to be loaded into the robot.

The difference between this type of loads into the robot is determined by WMS software using the cause field.

If an item must be managed by batch number and expiry date and in the item barcode there is no information about the batch on the pack(i.e. it is not a GS1 or HIBC barcode with the batch and the expiry date valued) in the refill list the item must appear only once and the batch number must be specified.

If the refill is not managed by the Customer IT Systems, it is possible to manage it using the WMS GUI.

It is exposed by the Core middleware to allow the maximum implementation flexibility from the Custom Middleware (for example the Custom Middleware can be called from the Customer IT system when a new refill list is confirmed, in this case the Custom Middleware will simply forward the call, or the Custom Middleware can periodically request the Customer IT system for new open orders and then call the service on the Core Middleware with these ones.

### Exposed by

Core Middleware

### URL

/middleware/refill

### Method

POST

### Input:

A JSON with the following structure is expected:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** | Default value |
| refill | Object | The refill list | Yes |  |
| code | string | Refill Code. Used in the service to transmit movements | Yes |  |
| cause | String | Cause code of this refill | No | TRREF |
| pickupCenter | string | The pickup center to associate to this load. Typically populated only by stock returns from wards. | No |  |
| date | Date | When the load to the robot is expected to happen | No |  |
| destination | String | Destination area | No | Riedl Phasys |
| warehouse | String | The warehouse where the refill should happen. Useful only if there are more warehouses managed by the installation | No |  |
| supplier | Object | The supplier if this is a load of a whole bill. | No |  |
| Code | String | The code of the supplier | Yes |  |
| Description | String | The description of the supplier | Yes |  |
| items | Object Array | The items to refill | Yes |  |
| itemCode | String | Item code to be loaded | Yes |  |
| quantity | Integer | The maximum number of unit of measure configured on the item to load | Yes |  |
| batchNumber | String | The batch number of the packs to load. To be used only if the item is managed by batch number and there is no batch in the barcode (i.e. GS1 or HIBC) | No |  |
| expiryDate | Date | The expiry date of the packs to load. To be used only if the item is managed by batch number and there is no batch in the barcode (i.e. GS1 or HIBC) | No |  |

Example:

{

"refill": {

"code": "1REF-4578",

"cause": "TRREF",

"date": "2018-09-07T11:00:00"

"destination": null,

"warehouse": "FAR",

"items": [{

"itemCode": " ",

"quantity": 200,

"batchNumber": "WE45U",

"expiryDate": "2020-01-31T00:00:00"

}, {

"itemCode": "ER4-1",

"quantity": 10,

"batchNumber": null,

"expiryDate": null

}

]

}

}

### Output:

HTTP Status Codes:

200 = Success

If the http status 200 is returned the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| deliveryNumber | string | The delivery number to put in the Riedl Phasys GUI to start loading | Yes |

Example:

{

"deliveryNumber": "89714"

}

## Warehouse Stock

This flow is used for inventory/recheck management and for refill of the robot if managed by the customer IT Systems.

This method returns, given a warehouse, the stock of tall the items (that should be managed by the Robot/WMS) in that warehouse

### Exposed by

Custom Middleware/Cutomer IT Systems

### URL

/custom/warehouseStock

### Method

GET

### Input:

The following http parameters are sent to the service:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| idWarehouse | string | The code of the warehouse whose stock is to be known | Yes |
| itemCode | string | The code of the item whose stock is to be known | No |

If the itemCode is not sent, the stock of all the items in the warehouse is expected

### Output:

HTTP Status Codes:

200 = Success

If the http status 200 is returned the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| stocks | Object Array | The list of the stocks returned | Yes |
| itemCode | string | The code of the item | Yes |
| quantity | Integer | The number of unit of measure in stock in the warehouse | Yes |
| batchNumber | string | The batch number. If the stock is managed by batch (see managedByBatch in the items service) | No |
| expiryDate | Date | The expiry date. If the stock is managed by batch (see managedByBatch in the items service) | No |

Example:

{

"stocks": [{

"itemCode": "AB12-1",

"quantity": 245,

"batchNumber": "QRR1",

"expiryDate": "2023-02-28T00:00:00"

}, {

"itemCode": "AB12-1",

"quantity": 140,

"batchNumber": "QRW3",

"expiryDate": "2023-07-31T00:00:00"

}

]

}

## Robot Stock

This method exposes the stock of the areas where the WMS is master of the stock, typically the robotic warehouse.

### Exposed by

Core Middleware

### URL

/middleware/stock

### Method

GET

### Input:

The following http parameters are sent to the service:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| idWarehouse | String | The code of the warehouse whose stock is to be known | Yes |
| itemCode | string | The code of the item whose stock is to be known | No |

If the itemCode is not sent, the stock of all the items is returned

### Output:

HTTP Status Codes:

200 = Success

If the http status 200 is returned the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| stocks | Object Array | The list of the stocks returned | Yes |
| itemCode | string | The code of the item | Yes |
| quantity | Integer | The number of unit of measure in stock in the warehouse | Yes |
| batchNumber | string | The batch number. If the stock is managed by batch (see managedByBatch in the items service) | No |
| expiryDate | Date | This is the expiry date if the stock is managed by batch (see managedByBatch in the items service) or the date where the stock should be rechecked in the robot to verify it is not expired otherwise | No |

Example:

{

"stocks": [{

"itemCode": "AB12-1",

"quantity": 245,

"batchNumber": "QRR1",

"expiryDate": "2023-02-28T00:00:00"

}, {

"itemCode": "AB12-1",

"quantity": 140,

"batchNumber": "QRW3",

"expiryDate": "2023-07-31T00:00:00"

}, {

"itemCode": "AB12-2",

"quantity": 75,

"batchNumber": null,

"expiryDate": "2019-07-01T00:00:00"

}

]

}

## Unload Requests

This method is used to ask to unload, when the unload does not start immediately. The unload requests are put in a buffer and then sent to the robot as soon as possible.

It is exposed by the Core middleware to allow the maximum implementation flexibility from the Custom Middleware (for example the Custom Middleware can be called from the Customer IT system when a new request is confirmed, in this case the Custom Middleware will simply forward the call, or the Custom Middleware can periodically request the Customer IT system for new confirmed unload requests and then call the service on the Core Middleware with these ones.

The requests are managed transactionally one by one (which means that a failure on a single request causes a full rollback for that request, the others will not be affected).

### Exposed by

Core Middleware

### URL

/middleware/requests

### Method

POST

### Input:

A JSON with the following structure is expected:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** | Default value |
| requests | Object Array | The requests list | Yes |  |
| requestCode | string | Code of the request in the Customer IT Systems. Used in the service to transmit movements | Yes |  |
| userCode | string | A string to show to the user that can help him identify the request in the Customer IT Systems | No |  |
| cause | String | Cause code of this unload | No | SCEVR |
| date | Date | Request date | No | Now |
| deliveryDate | Date | Maximum date by which the request must be downloaded (to increase the priority of an urgent request) | No |  |
| pickupCenterCode | String | Code of the pickup center to which the unloading is to be sent (but it is easier to manage the unloads if this is populated so it is strongly recommended to manage it) | No |  |
| warehouse | String | The warehouse where the request must be handled. Useful only if there are more warehouses managed by the installation | No |  |
| shipmentNote | String | The place where the goods must be shipped |  |  |
| urgent | Boolean | True the unload must be managed urgently, false otherwise | No | false |
| requestRows | Object Array | The items to unload | Yes |  |
| itemCode | String | The code of the item to unload | Yes |  |
| quantity | Integer | The number of units of measure to unload | Yes |  |
| batchNumber | String | The batch number of the packs to unload. It is recommended to let the robot choose what to unload using FEFO logic during regular use | No |  |
| rowRequestCode | String | Code of the line of the request in the Customer IT System. The couple (requestCode, rowRequestCode) must be unique | Yes |  |

Example:

{

"requests":[{

"requestCode": "1 50 JRIC 45098",

"userCode": "JRIC/45098",

"cause": "SCEVR",

"date": "2018-09-07T00:00:00",

"deliveryDate": "2018-09-10T00:00:00",

"pickupCenterCode": "1VFCAR",

"warehouse": "FAR",

"shipmentNotes":"First Floor, ask about Peter Parker"

"urgent": true,

"requestRows": [{

"itemCode": "AB12-1",

"quantity": 40,

"batchNumber": null,

"rowRequestCode": "1

},{

"itemCode": "ER4-1",

"quantity": 5,

"batchNumber": null,

"rowRequestCode": "2"

}]

},{

"requestCode": "1 50 JRIC 45108",

"userCode": "JRIC/45108",

"cause": "SCEVR",

"date": "2018-09-07T00:00:00",

"deliveryDate": null,

"pickupCenterCode": "1VFMED",

"warehouse": "FAR",

"urgent": false,

"requestRows": [{

"itemCode": "AB17-1",

"quantity": 15,

"batchNumber": null,

"rowRequestCode": "1"

},{

"itemCode": "ER9-1",

"quantity": 10,

"batchNumber": null,

"rowRequestCode": "2"

}]

}]

}

### Output:

HTTP Status Codes:

200 = Success

If the http status 200 is returned the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| requestsResponse | Object Array | The response of the operation for each request | Yes |
| requestCode | String | The code of the request | Yes |
| success | boolean | True means the request was created/modified, false that it is unchanged | Yes |
| errorMessage | string | The error message if success is false | No |

Example:

{

"requestsResponse":[{

"requestCode": "1 50 JRIC 45098",

"success": true,

"errorMessage": null

},

{

"requestCode": "1 50 JRIC 45108",

"success": false,

"errorMessage": "Unknown pickup center"

}]

}

## Direct Unload Request

This method is used to ask to unload, when the unloading must start immediately. This usually happens when there is an outpatient counter and the exit is supervised.

It is exposed by the Core middleware and it is strongly suggested that the Customer IT Systems simply call the service exposed from the Core Middleware to not introduced latency.

In the output the system returns which items the system thinks it can deliver, so that the customer IT Systems can direct the user to pick some item manually if needed/available.

Instruction that a pack must be opened and that a partial quantity must be conserved on the counter is also given.

The answer must not be used as mandatory, since it can change during the delivery (hardware problems, stock problems etc.)

### Exposed by

Core Middleware

### URL

/middleware/directRequest

### Method

POST

### Input:

A JSON with the following structure is expected:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** | Default value |
| request | Object | The request | Yes |  |
| requestCode | string | Code of the request in the Customer IT Systems. Used in the service to transmit movements | Yes |  |
| userCode | string | A string to show to the user that can help him identify the request in the Customer IT Systems | No |  |
| cause | String | Cause code of this unload | No | SCEVR |
| date | Date | Request date | No | Now |
| pickupCenterCode | String | Code of the pickup center to which the unloading has to be sent (but it is easier to manage the unloads if this is populated so it is strongly recommended to manage it) | No |  |
| warehouse | String | The warehouse where the request must be handled. Useful only if there are more warehouses managed by the installation | No |  |
| robotExit | Integer | The exit to which the items must be delivered | Yes |  |
| requestRows | Object Array | The items to unload | Yes |  |
| itemCode | String | The code of the item to unload | Yes |  |
| quantity | Integer | The number of units of measure to unload | Yes |  |
| batchNumber | String | The batch number of the packs to unload. It is recommended to let the robot choose what to unload using FEFO logic during regular use | No |  |
| rowRequestCode | String | Code of the line of the request in the Customer IT System. The couple (requestCode, rowRequestCode) must be unique | Yes |  |

Example

{

"request":{

"requestCode": "1 50 JRIC 45098",

"userCode": "JRIC/45098",

"cause": "SCEVR",

"date": "2018-09-07T00:00:00",

"pickupCenterCode": "1VFCAR",

"warehouse": "FAR",

"robotExit": 1,

"requestRows": [{

"itemCode": "AB12-1",

"quantity": 40,

"batchNumber": null,

"rowRequestCode": "1"

},{

"itemCode": "ER4-1",

"quantity": 5,

"batchNumber": null,

"rowRequestCode": "2"

}]

}

}

### Output:

HTTP Status Codes:

200 = Success

If the http status 200 is returned, the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| requestResponse | Object | The object containing the response | Yes |
| requestCode | String | The code of the request | Yes |
| requestRows | Object Array | The stock that should be unloaded | Yes |
| itemCode | string | The code of the item (if the required item is virtual it will be a different item than the required one) | Yes |
| quantity | Integer | The number of units of measure that will be moved | Yes |  |
| sign | Integer | +1: increase of quantity  -1: decrease of quantity  For example an unload will always has sign -1, a supplier bill +1, but a recheck can have bot +1 or -1 (if the physical quantity is greater or less than the theoretical quantity) |  |  |
| partialPackQuantity | Integer | If the required quantity is not divisible in full pack, the robot will unload a full pack and the Customer IT Systems must give to the user the instruction to open a pack and conserve this quantity | No |  |
| batchNumber | String | The batch of the items that will be unloaded if the stock of the item is managed by batch, null otherwise | No |  |
| expiryDate | Date | The expiry date of the items that will be unloaded if the stock of the item is managed by batch, null otherwise | No |  |
| rowRequestCode | String | Code of the line of the request in the Customer IT System. The couple (requestCode, rowRequestCode) must be unique | Yes |  |

Example:

{

"requestResponse":{

"requestCode": "1 50 JRIC 45098",

"requestRows": [{

"itemCode": "AB12-1",

"quantity": 40,

"partialPackQuantity": 5,

"batchNumber": "QrtE",

"expiryDate": "2019-02-28 00:00:00",

"rowRequestCode": "1"

},{

"itemCode": "ER4-1",

"quantity": 5,

"partialPackQuantity": null,

"batchNumber": null,

"expiryDate": null,

"rowRequestCode": "2"

}]

}

}

## Movements

This service is used to send the movements to the Customer IT services.

Both the movements required by the Customer IT services and the spontaneous ones use this service.

The movements are sent only when all the activity on a document (i.e. an unload request) is completed.

Example of movements are:

* Unload to fulfill a request by a pickup center
* Load of a bill into the Robot
* Unload for overtime unload
* Refill of the robot from external stock
* Movements for recheck

### Exposed by

Custom Middleware/Customer IT System

### URL

/custom/movements

### Method

POST

### Input:

A JSON with the following structure is sent:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** | Default value |
| documents | Object | The documents which contain the unload movements | Yes |  |
| requestCode | string | Code of the request in the Customer IT Systems. Null if the documents originates from the WMS | No |  |
| unloadCode | string | Code of the unload in the WMS | Yes |  |
| pickupCenterCode | String | Code of the pickup center | No |  |
| date | Date | Document date | Yes |  |
| cause | string | Cause of the unloading | Yes |  |
| note | string | A Note added to the unload document by the user | No |  |
| shipmentDetails | string | A string containing the details on where to ship the item (tipically just copied from the request) | No |  |
| deliveryNoteDate | Date | A date for the document | Yes |  |
| deliveryNote | String | The code of the delivery document in a bill. Used only if the cause is a LOAD one | No |  |
| warehouse | String | The warehouse of the document. Useful only if there is more than one warehouse. | Yes |  |
| movements | Object Array | List of movements | Yes |  |
| itemCode | String | The code of the item to unload | Yes |  |
| quantity | Integer | The number of units of measure to unload | Yes |  |
| batchNumber | String | The batch number of the packs to unload. | No |  |
| expiryDate | Date | The expiry date of the unloaded packs | No |  |
| serialNumbers | String Array | Array of the serial numbers of the unloaded packs read by the barcode scanners of the robot | No |  |
| rowRequestCode | String | Code of the line of the request in the Customer IT System. The couple (requestCode, rowRequestCode) must be unique | Yes |  |
| basket | String | Code of the basket containing the item | No |  |
| managedByRobot | Boolean | true if the goods were managed by the robot, false if they were managed manually |  |  |

Example

{

"documents": [{

"requestCode": "1 50 JRIC 45098",

"unloadCode": "12567",

"pickupCenterCode": "1VFCAR",

"date": "2018-09-07T00:00:00",

"cause": "SCEVR",

"note": "Will send additional packs when available",

"shipmentDetails": "Pavilion C, First Floor",

"deliveryNoteDate": null,

"deliveryNote": null,

"warehouse": "FAR",

"movements": [{

"itemCode": "AV-14",

"quantity": 14,

"batchNumber": "2E45",

"expiryDate": "2021-09-30T00:00:00",

"serialNumbers": ["127689", "127690"],

"rowRequestCode": "1",

"basket": "124",

"managedByRobot": true

}, {

"itemCode": "TA-PU-PU",

"quantity": 22,

"batchNumber": "T5432",

"expiryDate": "2020-12-31T00:00:00",

"serialNumbers": null,

"rowRequestCode": "2",

"basket": "124",

"managedByRobot": false

}

]

}

]

}

Output:

HTTP Status Codes:

200 = Success

If the http status 200 is returned the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| documentsResponse | Object Array | The response of the operation for each document | Yes |
| unloadCode | String | The code of the unloaded items | Yes |
| success | boolean | “True” means the unload was correctly acquired, false means error acquiring the unload | Yes |
| errorMessage | string | The error message if success is false | No |

Example:

{

"documentsResponse":[{

"unloadCode": "12567",

"success": true,

"errorMessage": null

}]

}

## Inventory Request

With this service the Customer IT Systems asks to the WMS software to perform an Inventory.

### Exposed by

Core Middleware

### URL

/middleware/inventoryRequest

### Method

POST

### Input:

A JSON with the following structure is expected:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** | **Default value** |
| request | Object | The request | Yes |  |
| requestCode | string | Code of the request in the Customer IT Systems. Used in the service to transmit the stock | Yes |  |
| userCode | string | A string to show to the user that can help him identify the request in the Customer IT Systems | No |  |
| cause | String | Cause code of this inventory | No | CONTR |
| date | Date | Request date | No | Now |
| warehouse | String | The warehouse where the inventory should be performed. Useful only if there are more warehouses managed by the installation | No |  |
| itemCodes | String Array | The items to check.  If null it means all items | No |  |

Example

{

"request":{

"requestCode": "FAR\_2018\_03",

"userCode": "2018/03",

"cause": "COINV",

"date": "2018-03-31T00:00:00",

"warehouse": "FAR",

" itemCodes ": ["PA-12","TA-PU-PU"]

}

}

### Output:

HTTP Status Codes:

200 = Success

If the http status 200 is returned, the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| requestResponse | Object | The object containing the response | Yes |
| success | boolean | True if all goes well, false otherwise | Yes |
| errorMessage | string | If success is false the error message | No |

Example:

{

"requestResponse":{

"success": true,

"errorMessage": null

}

## Real Stock

This service is used by the WMS to communicate the physical stock after a Inventory Request by the Customer IT Services. This is useful to certify the stock.

### Exposed by

Custom Middleware/Customer IT Systems

### URL

/custom/inventoryResponse

### Method

POST

### Input:

A JSON with the following structure is expected:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** | **Default value** |
| inventoryResponse | Object | The object containing the real stock | Yes |  |
| requestCode | string | Code of the request received by the Customer IT Systems. | Yes |  |
| cause | String | Cause code of this inventory | No | CONTR |
| date | Date | Inventory Date | No | Now |
| warehouse | String | The warehouse where the items were checked. Useful only if there are more warehouses managed by the installation | No |  |
| items | Object Array | The list of rechecked items | Yes |  |
| itemCode | string | Code of the checked item | Yes |  |
| batchNumber | string | the batch number of the physical stock. Populated only if the stock is managed by batch number | No |  |
| expiryDate | date | the date of the rechecked item. Populated only if the stock is managed by batch number | No |  |
| quantity | integer | The physical quantity |  |  |

Example

{

" inventoryResponse ":{

"requestCode": "FAR\_2018\_03",

"cause": "COINV",

"date": "2018-03-31T00:00:00",

"warehouse": "FAR",

"items ": [{

"itemCode": "PA-12"

"batchNumber": "WE345",

"expiryDate": "2019-01-31T00:00:00",

"quantity":45

},{

"itemCode": "PA-12"

"batchNumber": "WE389",

"expiryDate": "2020-07-31T00:00:00",

"quantity": 36

},{

"itemCode": "TA-PU-PU"

"batchNumber": "QQ34RT",

"expiryDate": "2019-06-30T00:00:00",

"quantity": 36

}]

}

}

### Output:

HTTP Status Codes:

200 = Success

If the http status 200 is returned, the body of the response will contain a JSON with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Mandatory** |
| requestResponse | Object | The object containing the response | Yes |
| success | boolean | True if all goes well, false otherwise | Yes |
| errorMessage | string | If success is false the error message | No |

Example:

{

"requestResponse":{

"success": true,

"errorMessage": null

}