# 19.10. mSupply legacy REST APIs

## What is it?

This document describes an application programming interface for communicating with mSupply.

## Basics

- Communication is over HTTP protocol
- All data is submitted and returned as JSON
- HTTPS supported- you should use it. If you want to restrict to only HTTPS, then block the HTTP port mSupply is using on your router.

## Authentication

- All requests must have standard HTTP basic authentication headers
- You **must** use SSL to secure your communications unless you want to tell the world your password.

## GET - Getting data from mSupply

### For all calls

- The basic syntax

```
http://server_ip_address/type/resource/id_of_resource
e.g. http://example.com/mobile/name/524
will retrieve the complete record for name whose ID=524
```

- or

```
http://server_ip_address/resource?field=blah,field2=blah2&sortby=+foo,-bar
e.g.
http://example.com/mobile/name?name=foo@,customer=true&sortby=+name
will retrieve all names whose name starts with "foo" and who are customers, sorted by name in ascending order (a to z)

Note that a "/" is also allowed before the "?" in the URL e.g.
http://example.com/mobile/name/?name=foo@,customer=true&sortby=+name
```

- Method: GET
- Note that all price data will be returned rounded to 2 decimal places.

### Paging

- If you want a range of items, pass an HTTP header named “range” with a “from” and “to” value separated with a hyphen. See the following example to find out the right format for the “range” header.
  - e.g. to return the first 15 items:

```
range  bytes=1-15
```

  - The server will reply with a header containing the item range and the total number of items found:

```
 items 1-15/568
```

### Query notes

- For string searches you can use “@” as a wildcard, as in the example above
  - ?name=@foo@ will return all names containing “foo”
  - ?name=foo@bah will return all names starting with “foo” and ending with “bah”
- Allowed query operators are:
  - =
  - # (not equal to)
  - >=
  - <=
  - <
  - >
- A query operator must be followed by a query value
- All resources returned are first filtered by the server to only return valid results for the store the user is logged in to.
  - Transactions: only transactions created in that store
  - Items: only items visible in that store
  - Customers (names): only names visible in that store
  - More on stores here
  - There is a full list of field names here but note that as of Wednesday 31 October 2012 they haven't been updated for mSupply v3.2

## Available resources

### mobile/Items

| **Resource name** | item |
|---|---|
| **Fields you can query** | any |
| **Returns by default JSON array containing:** | id<br>code<br>item_name<br>stock_on_hand_tot |

| | |
|---|---|
| **When a single item ID is requested** | department<br>category<br>category2<br>stock_on_hand<br>stock_available<br>An array with a key of “lines” containing pack_size, available_quantity, batch, expiry_date, sell_price |

**mobile/Transactions**

| | |
|---|---|
| **Resource name** | transaction |
| **Fields you can query** | any |
| **Returns a JSON array containing:** | id<br>name (i.e. the name of the customer/supplier the transaction is to/from)<br>entry_date<br>confirm_date<br>status (a two letter code denoting the status of the transaction)<br>comment (the user-entered comment) |
| **If you only request a single ID (e.g example.com/mobile/transaction/558** | A JSON object containing every field for that record |
| **Special case one:<br>Getting An ID to use when submitting a new transaction; submit example.com/mobile/transaction/new_id Returns:** | JSON object containing:<br>new_transaction_id |

**mobile/Names**

| | |
|---|---|
| **Resource name** | name (a name can be a customer, a supplier, a manufacturer or a donor or a combination of these) |
| **Fields you can query** | any |
| **Returns by default JSON array containing** | id<br>code<br>name<br>bill_address1 |

**mobile/Invoice**

| | |
|---|---|
| **Resource name:** | invoice |
| **Fields you can query** | none. Just call the ID number like this<br>`http://example.com/mobile/invoice/578` |

| | |
|---|---|
| **Returns by default JSON array containing** | id<br>name_id<br>name_name (The actual name of the customer or supplier)<br>total<br>their_ref<br>type<br>entry_date<br>confirm_date<br>comment<br>entered_by (username not ID)<br>store (name not ID)<br>hold<br>lines: contains an array containing the lines on the invoice. Each array row contains:<br>item_id<br>item_name<br>quantity (no. of individual items i.e. pack size x quantity in a pack)<br>stock:<br>contains an array of all stock lines used on the invoice, and also any other stock lines for items on the invoice whose quantity is greater than zero. This means that for a “normal” invoice line editing operation (where you are not changing the item) you already have all the information that you need without a further call to the REST server when a line is edited) |

**Example**


```
{
  "id" : 587,
  "name_id" : 79,
  "invoice_num" : 256,
  "status" : "fn",
  "total" : 824.5,
  "their_ref" : "GIZ05",
  "type" : "ci",
  "entry_date" : "2012-09-30",
  "confirm_date" : "2012-10-30",
  "name_name" : "ZIGO HOSPITAL",
  "lines" : [ {
    "id" : 6396,
    "item_key" : 973,
    "item_line_key" : 960,
    "item_name" : "CLOXACILLIN TABS 250MG BOT/1000",
    "quantity" : 5,
    "pack_size" : 1000,
    "sell_price" : 161.61,
    "price_extension" : 808.05
  }, {
    "id" : 6397,
    "item_key" : 120,
    "item_line_key" : 14,
    "item_name" : "ATENOLOL TABLETS 50MG BOT/100",
    "quantity" : 5,
    "pack_size" : 100,
    "sell_price" : 3.29,
    "price_extension" : 16.45
  } ],
  "stock" : [ {
    "id" : 22126,
    "quantity" : 162,
    "batch" : "AD 1002",
    "expiry_date" : "2014-09-30",
    "pack_size" : 1000,
    "sell_price" : 147.09
  }, {
    "id" : 22307,
    "quantity" : 12,
    "batch" : "3026844",
    "expiry_date" : "2014-03-30",
    "pack_size" : 1000,
    "sell_price" : 139.3
  } ]
}
```


### mobile/Stock

| | |
|---|---|
| **Resource name** | stock |
| **Fields you can query** | any in the **items** table or for all the items of a particular invoice with the field “trans_id” |

| | |
|---|---|
| **Typically, having located an item to add to an invoice you will want to locate the total stock available for that item** | `http://example.com/mobile/stock?ID=123 // stock for item.ID 123`<br><br>`http://example.com/mobile/stock?trans_id=123 // stock for transaction.ID 123` |
| **Returned: an array of match records containing the item ID and the total stock quanitity like this** | `[ {`<br>`  "id" : 5265,`<br>`  "quantity" : 30,`<br>`}, {`<br>`  "id" : 5275,`<br>`  "quantity" : 100,`<br>`}, {`<br>`  "id" : 5274,`<br>`  "quantity" : 50,`<br>`}`<br>`]` |

### Stocktakes

| | |
|---|---|
| **Resource name** | stocktake |
| **Fields you can query** | any |
| **Fields returned from a query** | ID<br>description<br>stocktake created date<br>status |
| **You can filter on type of stocktake by querying the status e.g.** | `http://example.com/mobile/stocktake?status=sg` |
| **Requesting a particular stocktake returns the following fields for the stocktake itself** | description<br>stocktake created date<br>stocktake entered date<br>status<br>comment<br>An array whose key is “lines” containing an these fields for each item in the stocktake:<br>item code<br>item name<br>quantity (the pack size is assumed to be one)<br>snapshot quantity |
| **Example** | the following request would return the information for the stocktake with an id of 123:<br>`http://example.com/mobile/stocktake/123` |
| **Special Case: Creating a stocktake** | First you need to get a stocktake id that you can use by requesting the resource “new_id”<br>`http://127.0.0.1:8081/stocktake/new_id` |

A note on creating stocktakes: the recommended process is:

1. Get a new stocktake id
2. Get items- this will also return the current stock on hand.
3. Display this, along with a column for the user to enter current stock on hand
4. When ready to save, post the data (below). Wasn't so hard!

### Categories

Use this resource to return lists of different types of categories in mSupply.

| | |
|---|---|
| **Resource name** | category |
| **Fields you can query** | any |
| **Fields returned from a query** | id - the id of this category (unique only among categories of the same level)<br>description - the descriptive name of the category<br>type - the category's type. Can be one of 1level1, 1level2, 1level3, 2 or 3<br>parent_id - the id of the category which is the parent of this one. 0 means the category has no parent |
| **You can filter on type of category by querying the type (which can take the values “item”, purchase_order“, “transaction” and “name”)** | The following query will return all the categories for items<br>`http://example.com/mobile/category?type=item` |

### catalogue/Catalogue Items

| | |
|---|---|
| **Resource name** | catalogueItem |
| **First URI segment** | catalogue |
| **Fields you can query** | any |
| **Fields returned from a query for each item** | id - the item's unique id code<br>Item - the item's code<br>item_name - the item's descriptive name<br>indic_price - the catalogue price of the item<br>description - the lowest level category (category 1, level 3) that the item belongs to<br>units - the units the item is provided in e.g. Amp (ampule) Tab (tablet), report_quantity - the pack size of the item in the catalogue, e.g. the following query would return all items with an id greater than -1 and a name beginning with “a” |

* You can filter on the category that items belong to by querying the category number and level. There are 3 categories (1-3) but please note that category 1 is hierarchical; it has 3 levels - level 1 is the parent, level 2 categories are children of level 1s and level 3 categories are children of level 2s. So, altogether you can query category1level1id, category1level2id, category1level3id, category2id and category3id. Note that all queries by category will return items that belong to that category and those which belong to all its child categories e.g. this query will return all items assigned to the level 2 category 1 which has an id of 4 **and all items belonging to category 1 level 3 categories that are children of it**:

| | |
|---|---|
| **Example** | `http://example.com/catalogue/catalogueItem?category1level2id=4` |

### Customer Stock History Items

| | |
|---|---|
| **Resource name** | customerstockhistory |
| **Fields you can query** | any |

| | |
|---|---|
| **Fields returned from a mulitple query** | id, date_entered |
| **Fields returned from a single id** | id, date_entered,stock_take_date, lines (array) |
| | The lines object array contains:<br>id (the id of the line, [name_s_h_line]id)<br>item_id (the id of the item, [name_s_h_line]item_id)<br>item_name (the name of the item - [item]item_name using [name_s_h_line]item_id)<br>item_code ([item]code using [name_s_h_line]item_id)<br>store_stock (the amount of stock (number of items not packs) the supplying store has of this item)<br>stock_on_hand (the stock on hand entered by the user, [name_s_h_line]stock_on_hand)<br>usage (this store's daily usage of this item. Calculated as (stock on hand last stocktake + stock received last stocktake - stock on hand this stocktake)/number of days between this and the previous stocktake. Use new [name_s_h_line]previous_stock_on_hand and [name_s_h_line]previous_received_from_us fields in the calculation)<br>comment (the line's comment - [name_s_h_line]comment) |

### Version

| | |
|---|---|
| **Resource name** | version |
| **Fields you can query** | none |
| **Returned** | the current mSupply mobile version |
| **Example:** | `http://example.com/mobile/version` |

### mobile/masterlist

| | |
|---|---|
| **Resource name** | masterlist |
| **Fields you can query** | name_id, type (the type of list to return) |
| **Returned** | JSON containing the items for the associated masterlist belonging to the name of the specified type |
| **Example** | `http://example.com/mobile/masterlist/name_id="123"&type=weborder` |

### Settings

| | |
|---|---|
| **Resource name** | settings |
| **Fields you can query** | none. Precisely. |
| **Returned** | A JSON object with three entities: timeout, name_id, name- This is the id and name of the customer or supplier or store |
| **Example** | `http://example.com/mobile/settings` |

### mobile/Purchase orders

| | |
|---|---|
| **Resource name** | po |
| **Fields you can query** | id |
| **Returned** | JSON containing the Purchase Order details for the ID you queried |

| **Example** | `http://example.com/mobile/po/id="123"` |
|---|---|

### Reports

| **Resource name** | report |
|---|---|
| **Query** | type with one of three options: type=currentstock or type=expiringstock or type=orderedvsreceived |
| **Query parameters** | type=currentstock: none<br>type=expiringstock&nummonths=xx where nummonths is the number of months in advance of the current date to compare expiry dates with<br>type=orderedvsreceived&from=yyyymmdd&to=yyyymmdd where the from and to dates in yymmdd format are entered. |
| **Returned** | JSON containing the “Successfully sent” or any errors encounters |
| **Example** | `http://example.com/mobile/report/type=currentstock` |

### Patients

| **Resource name** | patient |
|---|---|
| **Query parameters** | first_name: first name of patient<br>last_name: last name of patient<br>dob: patient date of birth (DD:MM:YY)<br>policy_number: patient insurance policy number |
| **Returned** | JSON array of patient objects matching query parameters:<br>`[{`<br>`"ID":,`<br>`"name",`<br>`"phone",`<br>`"customer",`<br>`"bill_address1",`<br>`"supplier",`<br>`"email",`<br>`"code",`<br>`"last",`<br>`"first",`<br>`"date_of_birth",`<br>`"type",`<br>`"manufacturer",`<br>`"bill_address3",`<br>`"bill_address4",`<br>`"bill_postal_zip_code",`<br>`"supplying_store_id",`<br>`"nameNotes": ["note1","note2", ...],`<br>`"nameInsuranceJoin": [{`<br>`"ID",`<br>`"insuranceProviderID",`<br>`"nameID",`<br>`"isActive",`<br>`"policyNumberFamily",`<br>`"policyNumberPerson",`<br>`"type",`<br>`"discountRate",`<br>`"expiryDate",`<br>`"policyNumberFull",`<br>`"enteredByID" }, ... ],`<br>`}, ... ]` |
| **Example** | `http://example.com/api/v4/patient?first_name=j@&last_name=doe&dob=01/01/1960` |

#### Prescribers

| | |
|---|---|
| **Resource name** | prescriber |
| **Query parameters** | first_name: first name of prescriber<br>last_name: last name of prescriber<br>code: prescriber code |
| **Returned** | JSON array of prescriber objects matching query parameters:<br>`[{`<br>`"ID",`<br>`"code",`<br>`"first_name",`<br>`"last_name",`<br>`"initials",`<br>`"registration_code",`<br>`"category",`<br>`"address1",`<br>`"address2",`<br>`"phone",`<br>`"mobile",`<br>`"email",`<br>`"female",`<br>`"active",`<br>`"store_ID"`<br>`}, ... ]` |
| **Example** | |

### The Stock Endpoint

This API is used by the mSupply stock web app.

# POST/PUT - Sending data to mSupply

- Must include authentication header
- Data in HTTP body as JSON
- Two types of POST/PUT: **New** and **Update**
- For **New** records:
  - Must use POST method without 'If-Match: *' header.
  - Append the id of the record to be created to the resource: e.g. POST http://example.com/mobile/transaction/134
- For **Updates** to an existing record:
  - Must use either POST method with 'If-Match: *' header or PUT method (headers ignored). e.g. PUT http://example.com/mobile/transaction/134

#### Transactions

- Resource name: transaction
- Create a transaction
  - Submit a JSON object containing:
    - new_transaction_id (you must have already requested this from the server - see

above )
- comment
- name_id
- lines (contains a JSON array:)
  - item_id
  - quantity
  - directions (new in v3.84)

◦ Payload.

```
{"id": 34592,"comment":"test 1","name_id":3,
   "lines":[
       {"item_id": 19697,"quantity":10,"directions":"1t tid uf"},
       {"item_id": 22845,"quantity":5,"directions":"Take 5 tablets
at 9 a.m. tomorrow"}
       ]
     }
```

◦ successful completion returns a JSON object with “invoice_num” and a number
◦ Assumed data: The following data is assumed and can not be submitted (will be ignored if you try)
- The transaction type: it's a customer invoice (“ci”)
- The store ID: it's the store you're logged in to.
- The user: it’s the user who is logged in.
- The entry_date: it's today.
- The confirm_date: it's today.
- The status: will always be “cn” (confirmed).

- Updating a transaction:
  ◦ for supplier invoice we are editing only hold status for now. So json payload would be:

```
{"id":1002,"hold": "true"}
```

- for customer invoice we can also update line quantity

```
{"id": 34592,"comment":"test 2",
   "lines":[
       {"transline_id": 19697,"quantity":20},
       {"transline_id": 22845,"quantity":10}
       ]
     }
```

- Delete transaction lines:

◦ `{"id": 34592,"comment":"test 2","lines":[<all_lines>]`

### Stocktakes

| **Resource name** | stocktake |
|---|---|
| **Method** | POST |
| **Submit a JSON object containing** | id (you must have already requested this from the server - see above ) |

| **Finalising a stocktake** | body: include an item “status” with value “fn” |
|---|---|
| **Assumed fields** | stock_take_created_date created_by_id finalised_by_id store_id |
| **example 1** | Submit a stocktake and finalise http://example.com/mobile/stocktake/123<br>`{"status": "fn", id:123, stock_take_date:2013-02-08, description:8/2/2013 Stocktake, Comment: "test", "lines":[ {"id": 123,"item_id":234,"item_line_id":345,"item_name";"test_item1";"snapshot_qty":300,"stock_take_qty":310}, {"id": 222,"item_id":232,"item_line_id":212,"item_name";"test_item2";"snapshot_qty":400,"stock_take_qty":300} ] }`<br>if the previous status was “sg”, the mSupply server will now create the appropriate inventory adjustments, as well as changing the status of the stocktake. |
| **example 2** | Submit a stocktake with status “sg” http://example.com/mobile/stocktake/123<br>`{"new_stocktake_id": 34592,"description":"stocktake test", "status":"sg", "lines":[ {"item_id": 19697,"stock_take_qty":100}, {"item_id": 22845,"stock_take_qty":80} ] }` |
| **example 3** | Finalise an existing stocktake http://example.com/mobile/stocktake/123<br>`{"status":"fn"}` |

### Customer Stock History

| **Resource name** | customerstockhistory |
|---|---|
| **Method** | POST |
| **Submit a JSON object containing an array with 3 fields** | id<br>requested_quantity<br>user_comment |

# DELETE - Delete Records

## Available resources

### Stocktake

- Deleting a stocktake
  - e.g. http://127.0.0.1:8081/stocktake/123
  - Method: DELETE
  - Returns a JSON object containing a Description field with the value “stock take id xxx is deleted.” where xxx is the id of the stocktake deleted.

### Invoice

- <add here>

### Item

- <add here>

## Errors

- Errors are returned as a JSON object with one item “error” whose text content is the error message
- e.g.

```
{"error":"we can't take you seriously while you insist on wearing that cardy"}
```

- Here is a list of error messages you might see. Most are self-explanatory:
  - “No search parameter specified”
  - “No query parameter specified”
  - “Invalid resource specified”
  - “Invalid sort field specified”
  - “Invalid range header specified”
  - “No transaction with that ID found”
  - “Invalid transaction ID specified” (Different to above, in that you specified a non-numeric or negative ID)
  - “Yikes! Multiple transactions with same ID found” (you'll never see this error, or we'll be eating hats).

*Previous: **19.08. Stock web app*** | | *Next: **19.11. mSupply sync API***