ECOMMERCE AND

# Inventory Management

CSC 366 - Spring 2013

Shopatron

# Introduction

Shopatron is an eCommerce order management service. Specifically, Shopatron operates in the multi-tenant, software-as-a-service space. We are the number one cloud based order management tool for what we call Allied Commerce. We enable allied commerce by bringing together many different players to enable the order fulfillment that the consumer wants. Our vision is Any Product, Anywhere, Any Time; we bring together all the pieces necessary to make that happen.

The basics of eCommerce are already familiar to most, but some of the specifics of the Shopatron solution are likely new to the class. Especially relevant is the concept of multi-tenancy. In a multi-tenant solution, many different customers share the same resources – computer networks, servers, software, files and databases. The key to multi-tenancy is that these customers share the space without realizing they are "neighbors" with many other clients like them. In the case of Shopatron, we have clients that operate within the same industries and keeping the individual client boundaries up is essential to building client trust. Equally essential is the need to drive improved margins and simplicity of system management by allowing us to share resources through careful database and software design. The true magic of multi-tenancy is that by pooling clients together, an individual client receives more resources than they would on their own while also reducing the number of moving parts for the internal team to manage.

The core of Shopatron's service is order management. Put simply, we connect an order placed on a website to the entities that can fulfill the order. There are many modes of operation in determining how to route the order. This quarter, we will be focusing on one way the route can be determined – by knowing who has which products, and how many of those products are available at a specific location.

We will not focus on the wider order management topics of order management or fulfillment tracking. This quarter we will build an inventory management service similar to the one that powers part of the Shopatron solution. We will expose a web service as the primary way of managing and interacting with the data. We will also build a bulk loading process to manipulate inventory positions to create inventory data and to adjust the quantities available.

## Definitions

There are several terms that will be used throughout this document. The basic definitions are as follows:

**Available Inventory –** The quantity of items unallocated at a particular location.
**Allocation –** The act of dedicating one or more items for fulfillment, thereby removing the items from the available pool.
**Bin –** A named location within a store location. A bin is used as an organization tool and may refer to locations on a shelf in an isle in a warehouse to locations on a show room floor. All locations have one or more bins, and a default bin is used when a specific location does not further organize its products into individual bins.
**LTD –** Lifetime to date. This is a measure of how quickly or slowly a particular item is selling at a store location. It is very important to move certain items quickly in some industries. Prioritizing by allocations by LTD value helps promote moving items from the slower moving locations. The specific value of LTD

varies from client to client but is can always be used in relative comparison between locations of a single client.
**Location –** A physical location where products are kept, either a storefront or a warehouse.
**Manufacturer/Catalog –** A pair of id numbers that identify the client. A client will have one or more store locations in inventory.
**On-hand Inventory –** The count of products present at a specific location or bin, regardless of the quantity of products allocated.
**Safety Stock –** A limit on how many items can be safely allocated at a specific location. The safety stock represents "play" in the network where transactions can take time to propagate back to all players and during the propagation time, the available stock levels may not be accurate.
**SKU –** Stock keeping unit. This is an identifier assigned by the retailer to identify the product within their system. The SKU must identify a complete product, including the specific values of size or color or other options that apply to the type of product.
**Store –** A location where products are kept. It is synonymous with Location.
**Store-SKU –** A specific product SKU available at a specific store, possibly spread across more than one bin. Store-SKU combinations are used as basic units of system usage for billing purposes.
**UPC –** Universal Product Code. This is an identifier that is similar to SKU, except that the original product manufacturer assigns the UPC. A given product will have both a SKU and UPC, and the values may be different. If multiple retailers carry the same product, those products will likely have different SKUs but still have the same UPC.

## Interaction Overview

The lifecycle of an inventory record follows the steps outlined here. Each of these steps are expanded in the use cases in the next section.

1. Define the store locations.
2. Define the bins with locations.
3. Create the inventory position. A client uploads an inventory record for a specific bin within a location.
4. Check the available inventory for a given set of products.
5. Allocate the inventory to reserve for fulfillment.
6. De-allocate the inventory based on fulfillment completion.
7. Update on-hand inventory based on inventory updates.

# Use Cases

## Define Store Locations

For the purposes of this class, we will take a simplistic view of store locations. We will only create locations and leave out any maintenance of the location data once it has been created.

**Data Definition**
A store location has the following attributes:

- Name
- Fulfiller Id (Internal identifier of the retail chain)
- External Fulfiller Location Id (Client definied id for the location)
- Type (Store, Warehouse, Distribution Center, etc)
- Latitude
- Longitude
- Status (Active, Inactive)
- Safety Stock limit (The default safety stock level for all SKUs managed at this location)
- Manufacturer id
- Catalog id

Usage case:

1. Take bulk file of store locations, defined as a CSV.
2. For each row in the CSV file, send the data to the create store location API. The store location API will create the store location with a single bin of “default”.

## Define Store Bins

For the purposes of this class, we will take a simplistic view of the store bins as well. We will only create bins within store locations. All further maintenance of the bin data will be ignored at the beginning. Each bin will have a name, but the name “default” is reserved.

Usage case:

1. Take bulk file of bins, defined as a CSV. The bin file will link external fulfiller location ids to the name of bins.
2. For each row in the CSV file, send the data to the create store location bin API.

## Bulk Inventory Update

This is one of the main cases for managing inventory. The bulk inventory case will create and/or update inventory records for bins within store locations. There are two primary subcases to look at, creating a new inventory record or updating a inventory record. In either scenario, the same process is used but the update case may be more complicated.

An inventory record is tied to a specific bin within a specific store location. The inventory record has the product name, SKU, and UPC labels. In addition, all inventory records includes a safe stock value (possibly an override of the store default), and an LTD value. It also identifies the manufacturer and catalog ids as a cross check against the specific store location. Finally, the main purpose of the inventory record update is to set the inventory on hand count.

Usage case:

1. Take a bulk file of inventory data, defined as a CSV.
2. For each row, set the on hand count defined in the CSV by sending the information to the inventory update API. If the inventory record is new, the allocated count should start at 0.

## Trickle Inventory Update

The trickle inventory update is used to communicate when changes to the on hand inventory happens throughout the day. The idea is that each store location will feed updates back to the inventory system changes as they happen throughout the day. Changes could be selling items in store, loosing items in a store, destroying an item in store, shipping an item, receiving an new case of items, or any number of other changes.

The key difference between a trickle update and an inventory update is that a trickle update is an increment. The increment is either a positive or negative value that needs to be reflected back to the on hand inventory counts.

Usage case:

1. Take a trickle feed of inventory data, defined as a CSV.
2. For each row, send through the external fulfiller location id, manufacturer id, catalog id, bin name, SKU, UPC, and on hand delta to the trickle update API.

## Get Inventory

The get inventory call is the heart of the service. There are several modes of using the get inventory call, described here. All inquiries to the inventory system revolve around determining how much product is available.

There are a few use modes for the Get Inventory request. However, there are some common elements to all requests. All get inventory requests involve:

- A manufacturer id and catalog id
- One or more SKU/UPC combinations, with a minimum on-hand stock threshold.
- Limited to one type of store location.
- An upper limit on how many locations should be returned.
- A flag on if safety stock should be enforced.
- A flag on if negative available inventory should be included.
- A flag to order by LTD or by quantity available.

The variations of the Get Inventory request are:

1. Limit the results to one or more locations, identified by name.
2. Limit the results to locations within a specific distance radius of a given Latitude and Longitude.
3. Limit the results to locations within a specific distance radius of a given zip code.

The locations returned always are filtered by the manufacturer id and catalog id. The locations returned also always return the list of bins within the location that contain the requested product.

The results returned should be ordered by the specified value, descending. The order should prefer locations that have all specified items available first. Partial match locations should be listed second.

## Allocate Inventory

Allocations are used to hold inventory for orders being processed. They reserve a specific quantity of products for an indefinite time. Once an allocation has been made, the number of available products is decremented by the allocated count. Allocations are always increments and not absolute allocation counts. Allocations are made against a specific store location's bin.

Usage case:

1. (pre-condition) Typically, a get inventory call is made to determine what locations and bins are available.
2. The allocation API is called with a list of SKU/UPCs and specific store locations and bins.
3. The SKUs are all allocated, or the allocation fails due to insufficient available inventory.

## De-allocate Inventory

De-allocation is the companion to the allocation call. There are two primary reasons to use a de-allocation. First, when an item will now be fulfilled from a different location. In this case, we simply decrease the allocation count and increase the on-hand count. The second case is for when we fulfill an order. In this case, the allocation count is decreased and the on-hand count is decreased.

An error should be thrown if a de-allocation is requested when the allocated count is not great enough to correspond to the de-allocation request.

# Extra Use Cases

There are a few extra problems that some of the groups in class will feel challenged to tackle. We will dig into more details on the cases as groups become available to work on them.

## Calculate LTD Automatically

In theory, the LTD value can be calculated by watching the adjustments to the on hand counts over time. There are a few assumptions that need to be made. For example, you will need to assume the age of products when they move to a specific location. The idea of this task will be to calculate a relative ranking of inventory data to prioritize the slower moving changes over locations that sell more volume of the items.

## Web UI

There is a lot of data to deal with in the inventory system. In the base solution set defined in this document, we are only interacting with the data by way of API. However, there is potential to build a rich interface based on an enhanced set of APIs.

# Technical Requirements

The following are the technical specifications for the successful completion of the inventory service:

- The service response time must be sub-second for get inventory and allocation calls.
- The service must complete allocations within a single transaction. Specifically, all allocations applied within a single AIP call must be committed or rolled back.
- The service must enforce multi-tenancy by only returning locations associated with the provided manufacturer id and catalog id.

# Deliverables

For CSC 366, the deliverables required for each group includes:

- A working database schema that models the data described in this document and in any follow-up question and answer sessions.
- An application that works with your database design to implement the application behavior described in the API calls.
- An implementation of the WSDL provided to class, describing the API calls that your application must conform to. In addition to the API methods described in this document, the WSDL may call for some simple data retrieval methods. These additional methods are expected to be self-explanatory.
- In addition to the main application, a few bulk loaders will need to be built. The bulk loaders will work with CSV files and make API calls to load the data into the application.