Cookbook - REST API basics

- Overview
- Basic REST call flow
- Flows overview
  - Customer account flows
    - Profile flow
    - Address book management flow
    - Wishlists flow
    - Recently viewed flow
    - Order history
  - Browsing, search and navigation flows
    - Category navigation flow
    - Content navigation flow
    - Product view flows
    - Search and navigation flow
  - Checkout flow
- Resources

Overview

> This cookbook was written using 3.0.0 API. Most concepts are valid, however please ensure that you refer to the latest swagger spec f or the right API depicted in the flowcharts outlined in this document.

REST API is facilitated by Spring MVC where controllers are mapped to XML and JSON view resolvers. The choice of whether to use XML or JSON in request and/or response body is controlled via request headers: **Content-Type** for request body type and **Accept** for the response body type. Currently only XML and JSON modes are supported with "application/xml" and "application/json" values respectively. Request and response type in all REST API calls are independent of each other, so it is possible to send JSON request and receive XML or JSON response and send an XML request and receive XML or JSON response - it all depends on the values in "Content-Type" and "Accept" headers sent with the request.

Customer session is maintained using token and/or cookies headers. Token is specified using **yc** header, whereas cookie is specified by **yc** cooki e. Note that token is given higher priority and is examined first. Each REST API call response contains both the header and the cookie. Third party applications must provide this token and/or cookie for every subsequent call to maintain customer session.

REST API is represented by a number of controllers each of which is focused on particular functional area:

Controller	Functional area
AuthenticationController	Login, logout and login state check
CartController	Viewing cart, mutating cart and wishlist and checkout (including checkout options)
CategoryController	Category navigation
ContentController	Content navigation and viewing
CustomerController	Viewing and updating profile, managing address book (including country options), viewing wish lists, viewing recently viewed items, viewing order history
NodeController	Viewing basic node information
ProductController	Viewing product and SKUs, viewing associations and tracking product views
SearchController	Product search and navigation

We strongly advise to refer to javadoc for controllers which show request parameters and depict example responses.

Basic REST call flow

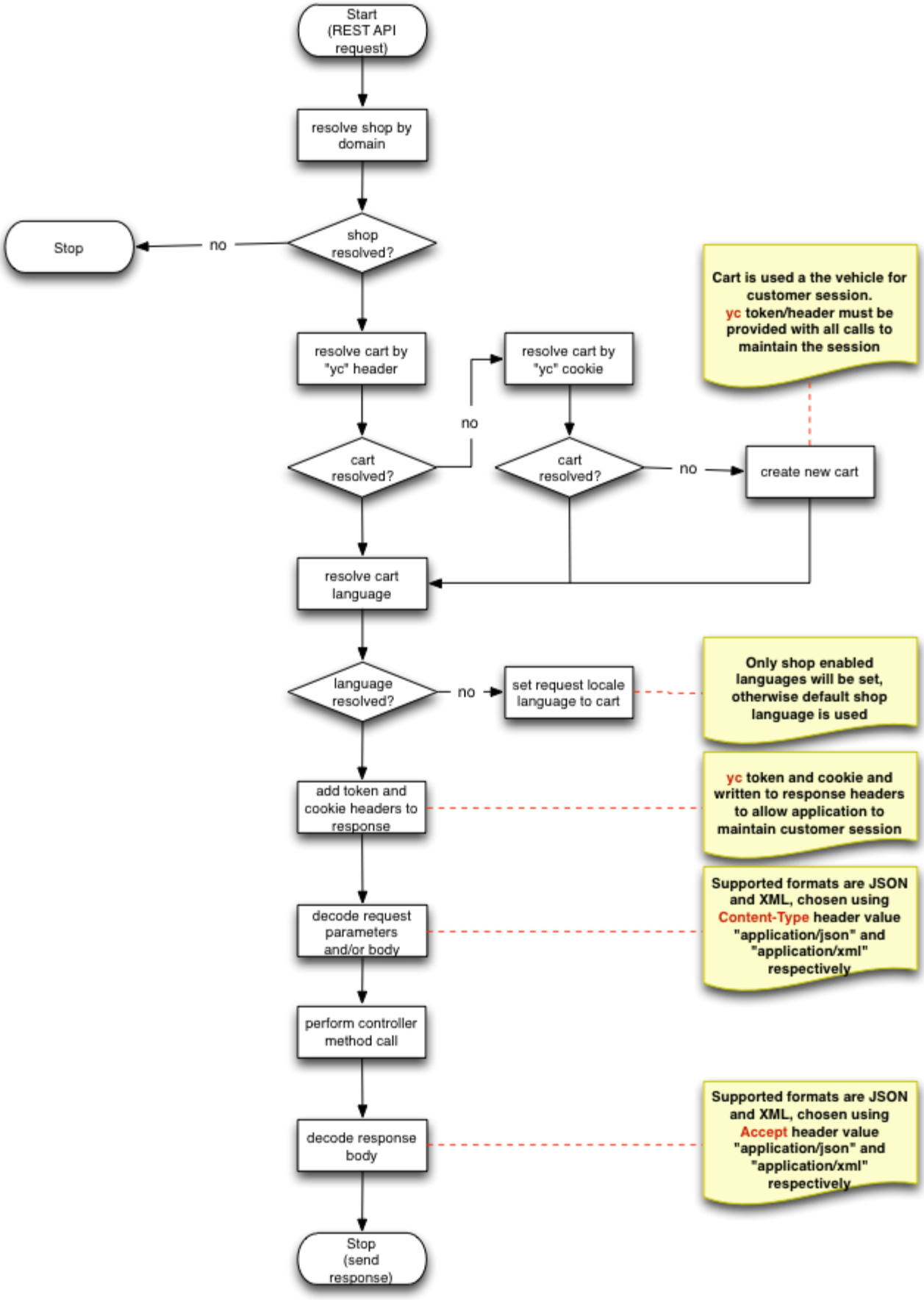
Basic REST call first comes through three stages of web filters:

1. shop resolution by domain (as per shop instance URL mapping)
2. cart resolution by token or cookie (at this point yc header and yc cookie are written to the response)
3. language resolution either from cart or from request (provided that language is enabled for given shop)

Next parameters and request body are decoded based on the Content-Type header.

Appropriate controller is invoked and model object is created.

After this the model object is decoded based on the Accept header and sent to response.


Start (REST API request)
resolve shop by domain
shop resolved?
no
Stop
resolve cart by "yc" header
resolve cart by "yc" cookie
no
cart resolved?
cart resolved?
no
create new cart
Cart is used a the vehicle for customer session. yc token/header must be provided with all calls to maintain the session
resolve cart language
language resolved?
no
set request locale language to cart
Only shop enabled languages will be set, otherwise default shop language is used
add token and cookie headers to response
yc token and cookie and written to response headers to allow application to maintain customer session
decode request parameters and/or body
Supported formats are JSON and XML, chosen using Content-Type header value "application/json" and "application/xml" respectively
perform controller method call
decode response body
Supported formats are JSON and XML, chosen using Accept header value "application/json" and "application/xml" respectively
Stop (send response)

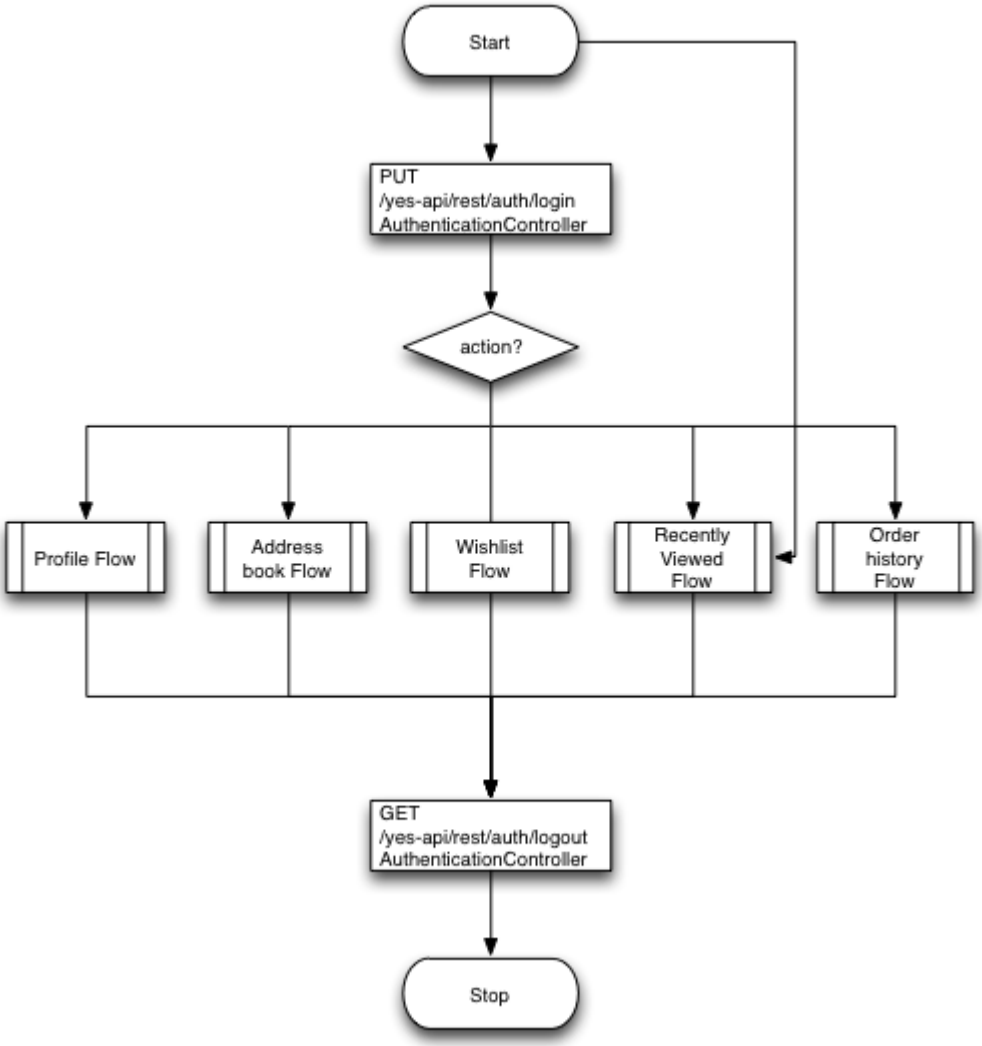
Flows overview

In this section we discuss basic functional flows detailing sequence of API calls that third party systems (e.g. mobile apps) would use for some of the customer journey use cases.

Customer account flows

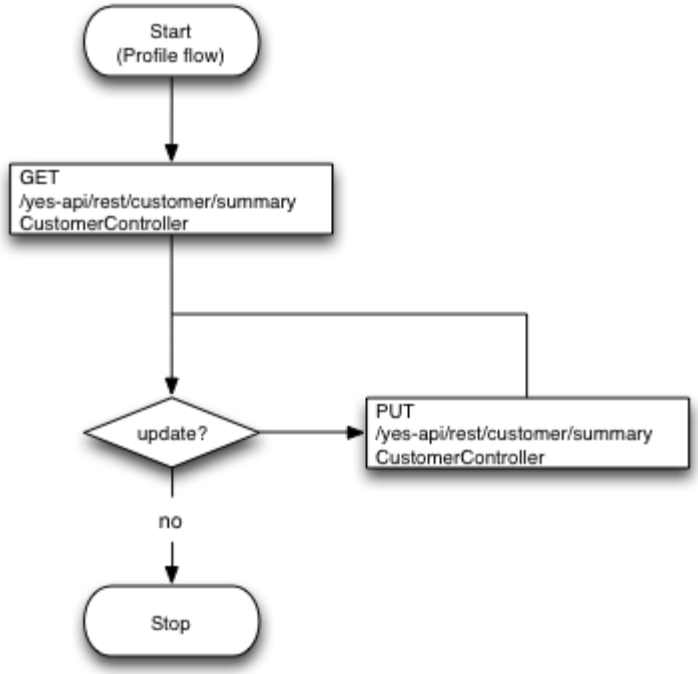
Customer account flows encompass several use cases: profile information view and update, address book management, wish lists view and update, recently viewed items and order history.

All customer account related flows apart from recently viewed items require a logged in customer session.



Profile flow

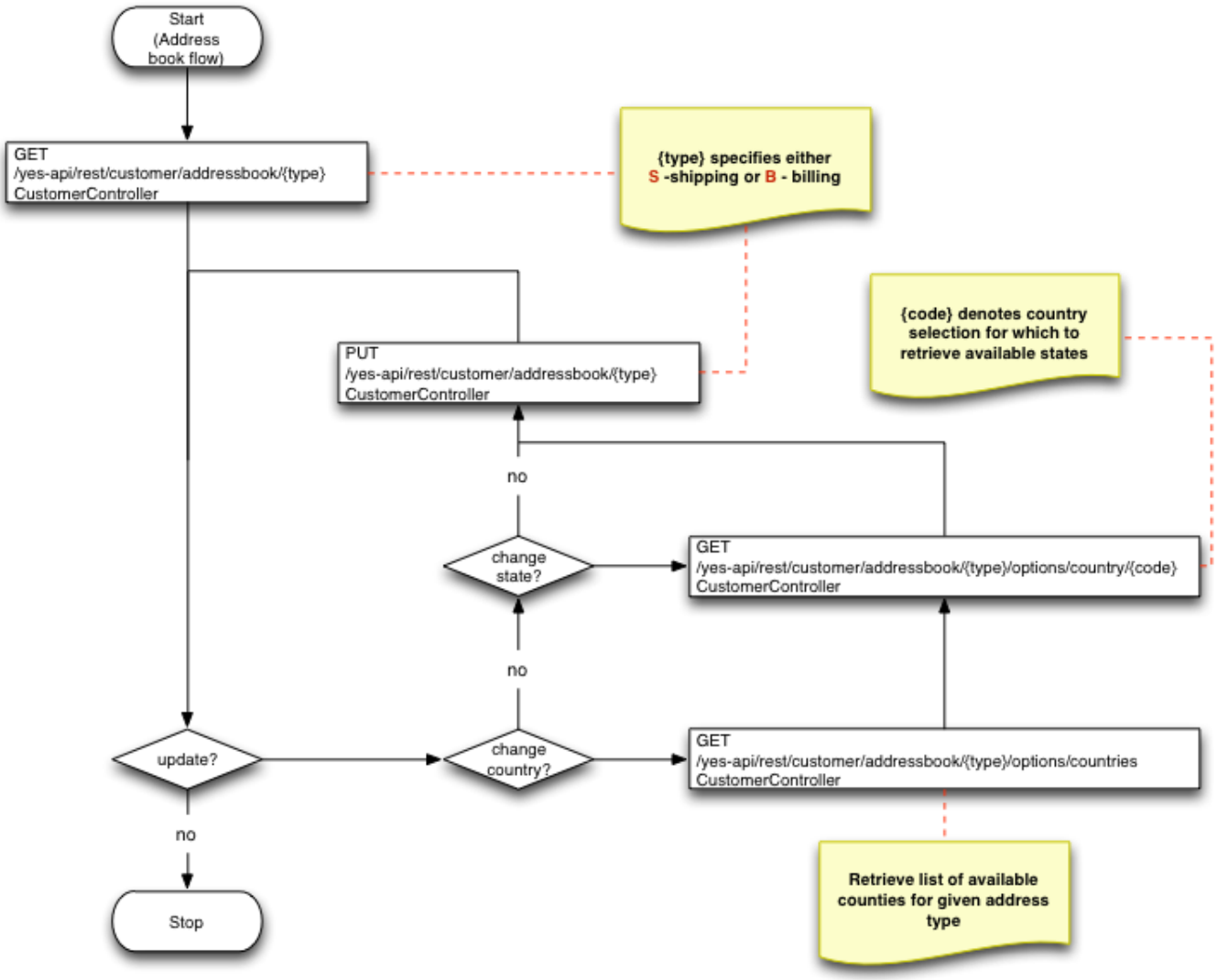
Profile summary interface allows to view and update basic profile information and additional information for shop configured customer profile attributes.



Address book management flow

Address book interface allows to view list of addresses by type (**S** for shipping and **B** for billing). Note that addresses are pre-filtered by shop allowed countries for shipping and billing addresses.

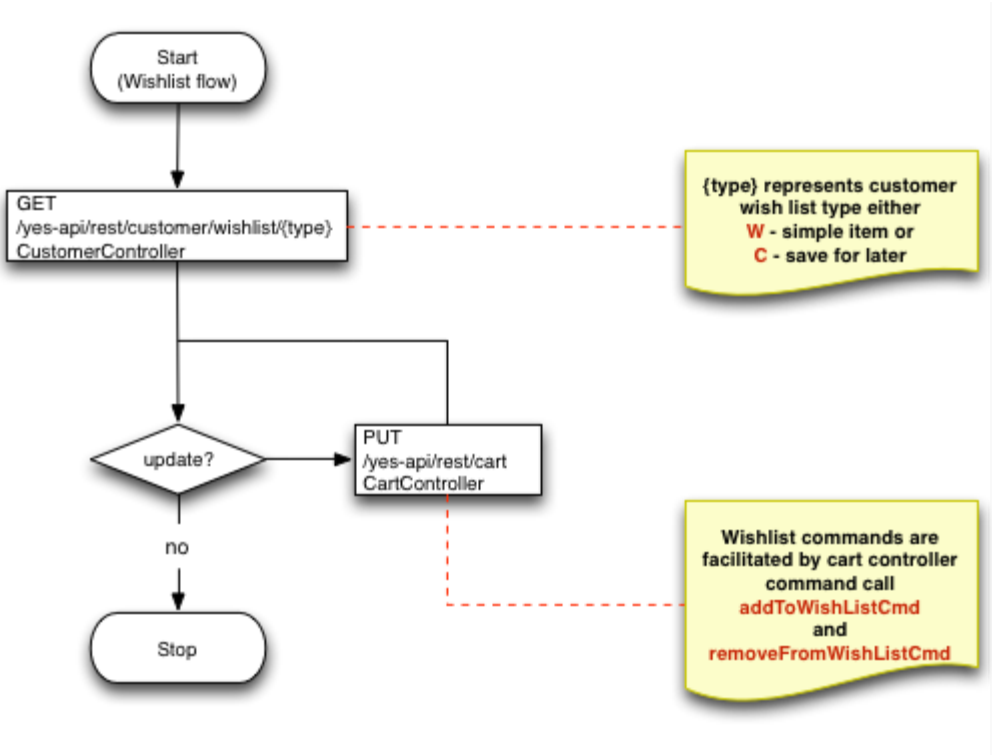
Since address objects rely upon location (country and state) this interface provides options for available countries per address type and states of particular country.



Wishlists flow

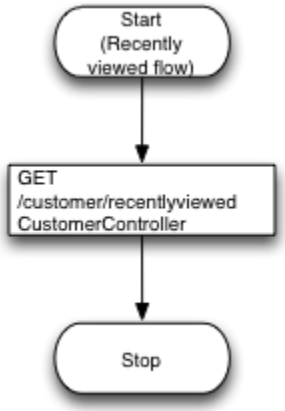
Wishlists interface allows to view wish lists by type. Currently two types are supported: **W** for basic wishlist and **C** for save for later cart items. W wish list can be further broken down by tags if multiple wish lists need to be supported.

Note that updating wish list items of all types is done via cart controller command interface since some operations with wish list directly influence cart state. Use **addToWishListCmd** and **removeFromWishListCmd** commands to add and remove wish list items respectively. For more details on parameters to these commands refer to ShoppingCartCommand.



Recently viewed flow

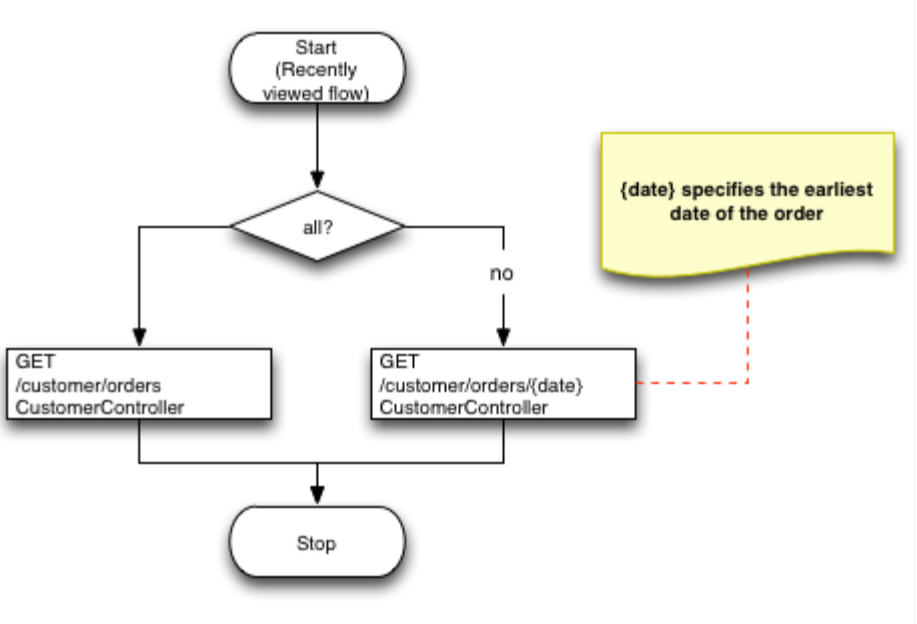
Recently viewed interface support both authenticated and anonymous customers and allows to view product view history in current session. The tracking on the product views is automatic and is done every time product is viewed using **GET /product/{id}** interface.



Order history

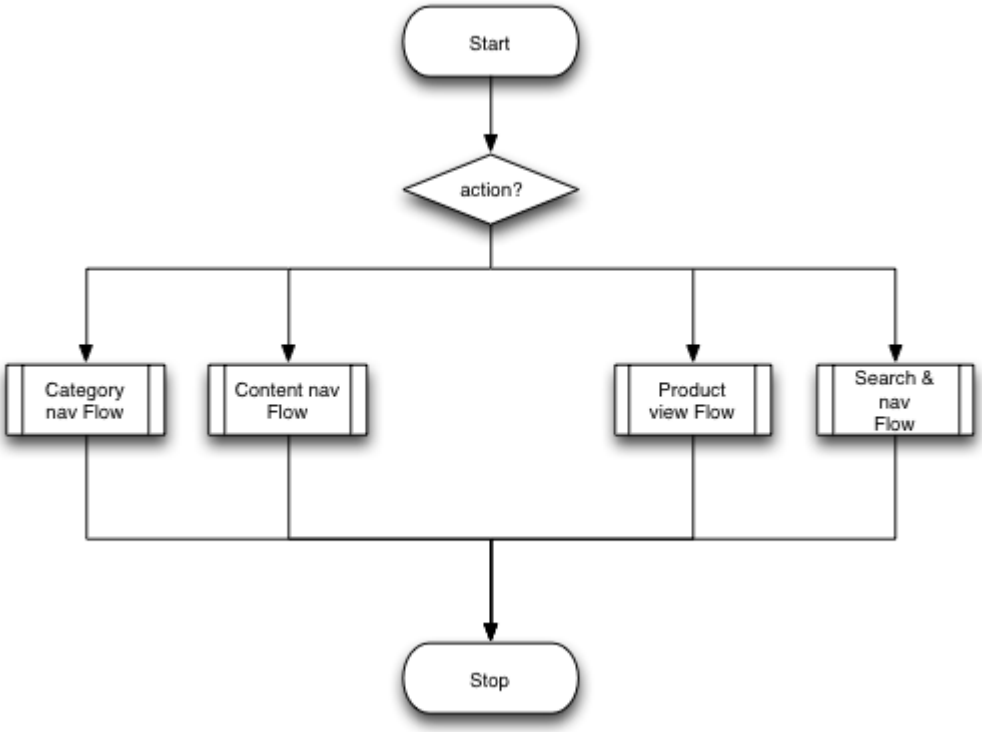
Order history interface allows either to view full order history or order history since some date.

For performance reasons it is highly advisable to use the "dated" interface to prevent full order history to be shown by default.



Browsing, search and navigation flows

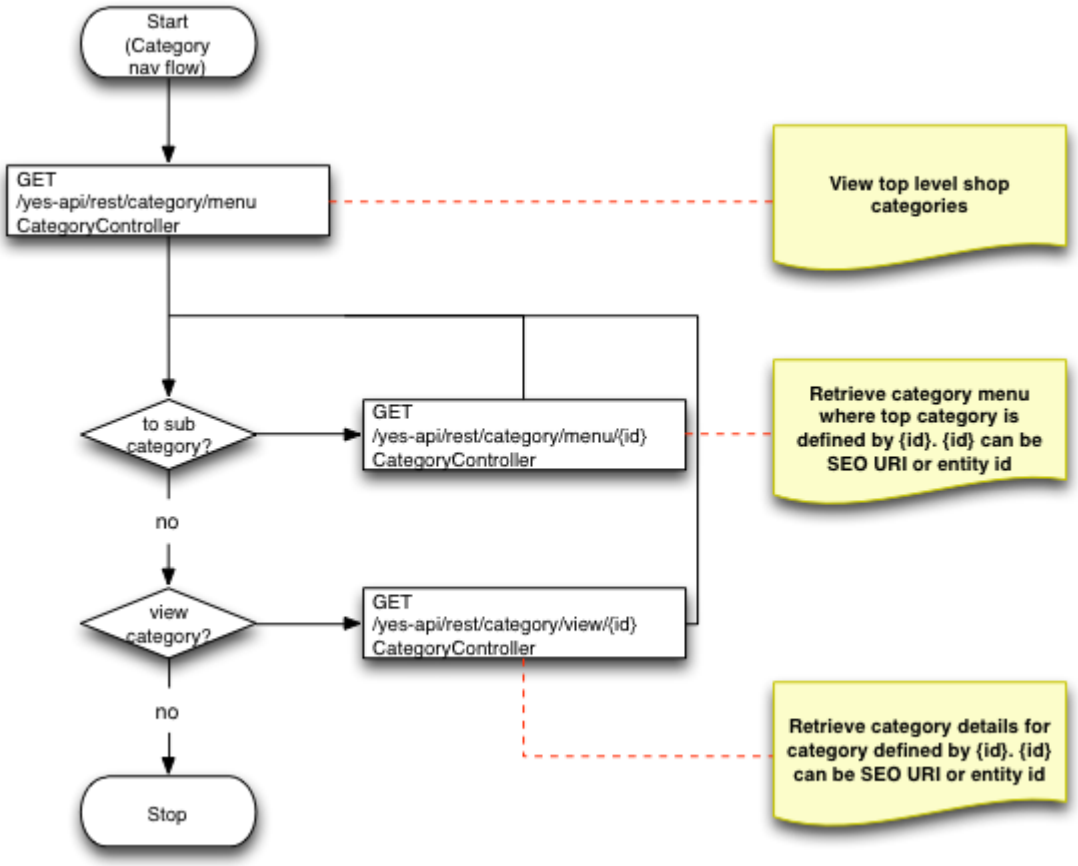
Browsing, search and navigation flows are all about navigating on the website, searching and viewing products as well as viewing content.



Category navigation flow

Category navigation flows allow to retrieve current category menu including breadcrumbs information, as well as retrieve detailed information about a specific category.

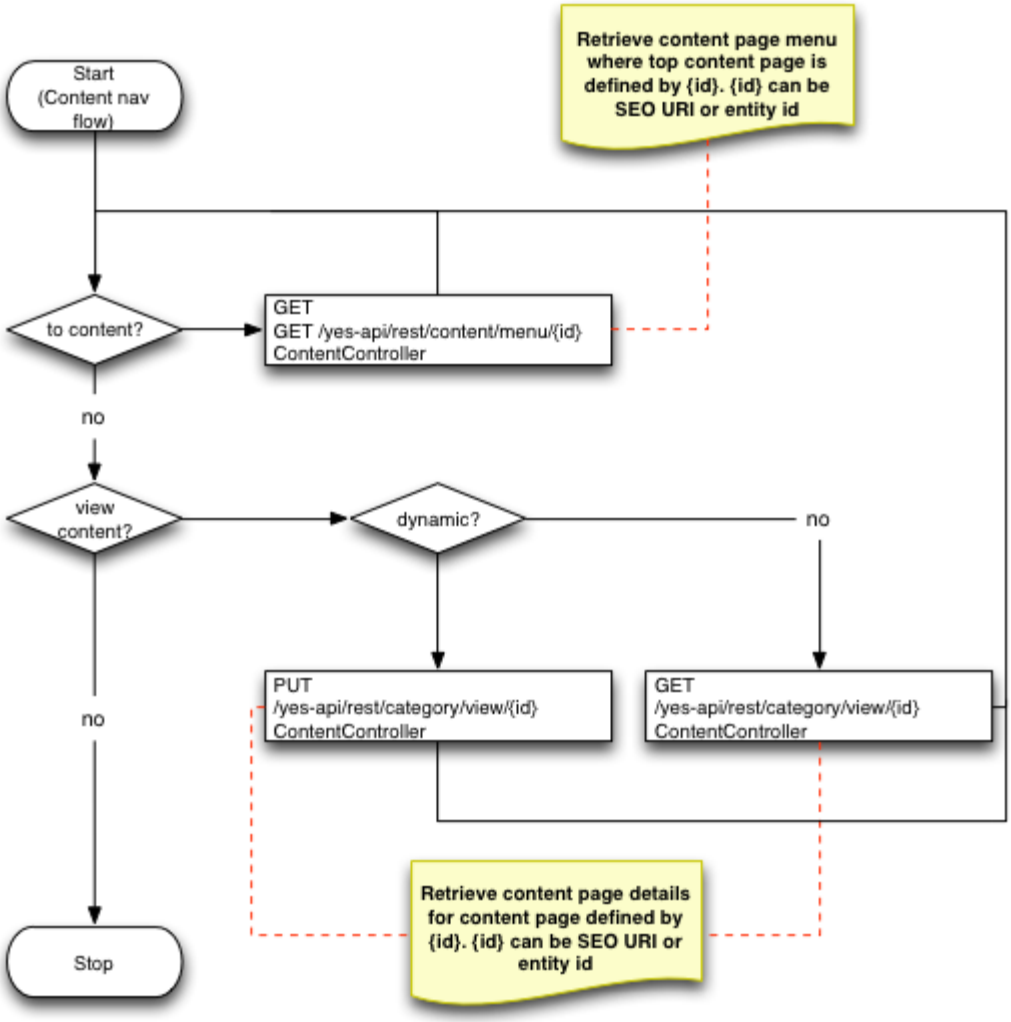
Effectively these calls translate to "category menu" and "breadcrumbs" on the web site.



Content navigation flow

Content navigation flows allow to retrieve current content page menu including breadcrumbs information, as well as retrieve detailed information about a specific content page.

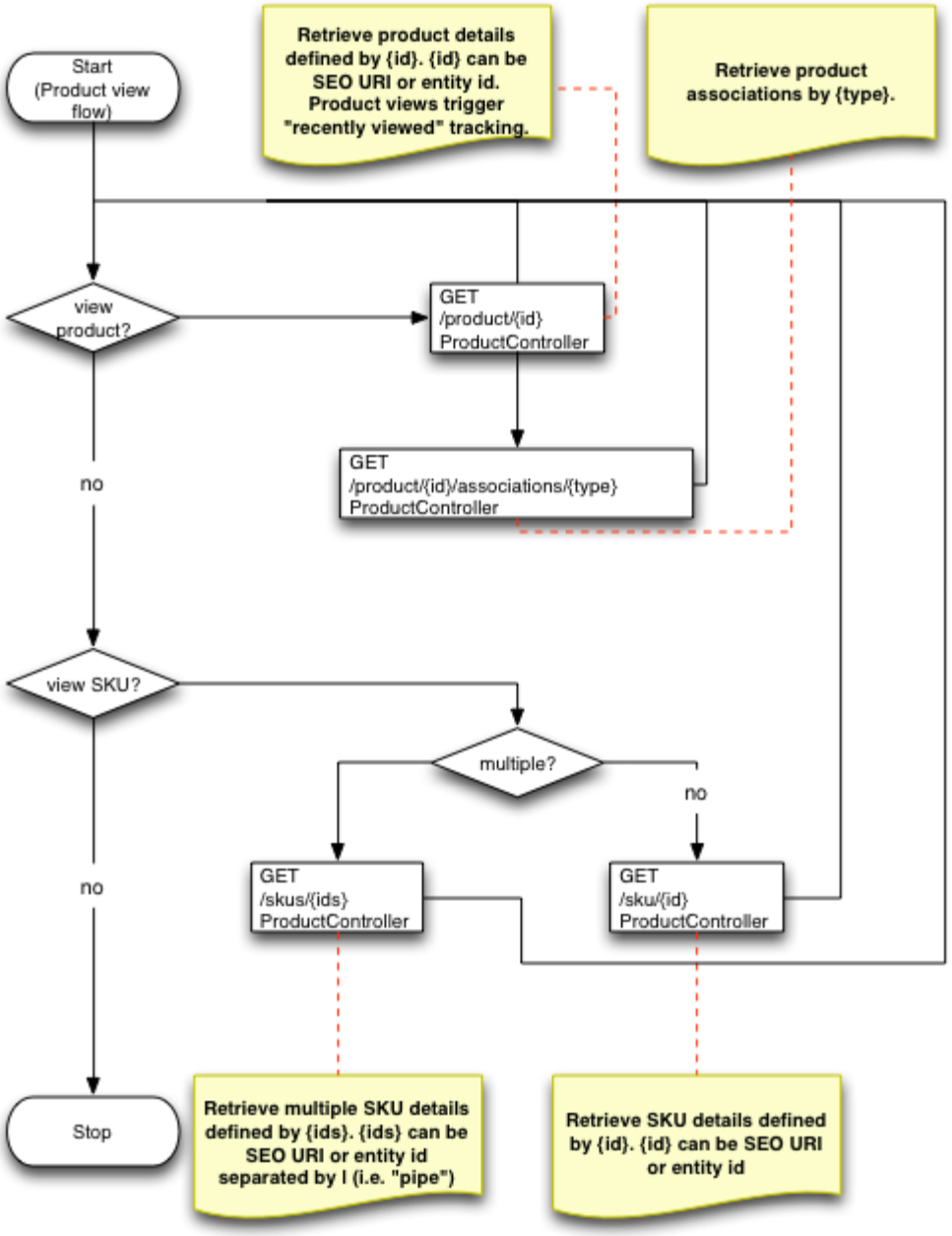
Effectively these calls translate to "content menu", "breadcrumbs" and "content page" on the web site.



Product view flows

Product view flows allow to retrieve product details, product association, single SKU details and multiple SKU details.

Effectively these calls translate to "product details page" and its components on the web site.

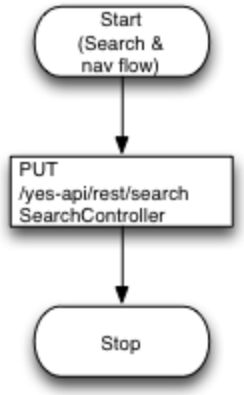


Search and navigation flow

Although the diagram of this flow is very simplistic this is the most complicated controller as it builds a very complex response model.

Search uses PUT call which expects a search request object that defines the parameters of the search such as: current category, whether to include filtered navigation options in response, search terms and applied filtered navigation options, pagination and sorting.

Response for the search call is a single page of products results with additional information about filtered navigation, available sorting and paging options and total results count.



Checkout flow

Checkout flow is all about populating the shopping cart and then placing an order.

Cart mutation processes such as adding/removing/updating products, adding/removing coupons and editing order message are all facilitated by **PUT /yes-api/rest/cart**command interface. For more details on parameters to these commands refer to ShoppingCartCommand.

When customer is ready to checkout there are four steps that they pass through.

Checkout step 1: login that allows existing customer to login or for new customers to register.

Checkout step 2: choose address where customer selects the shipping and billing address they wish to use.

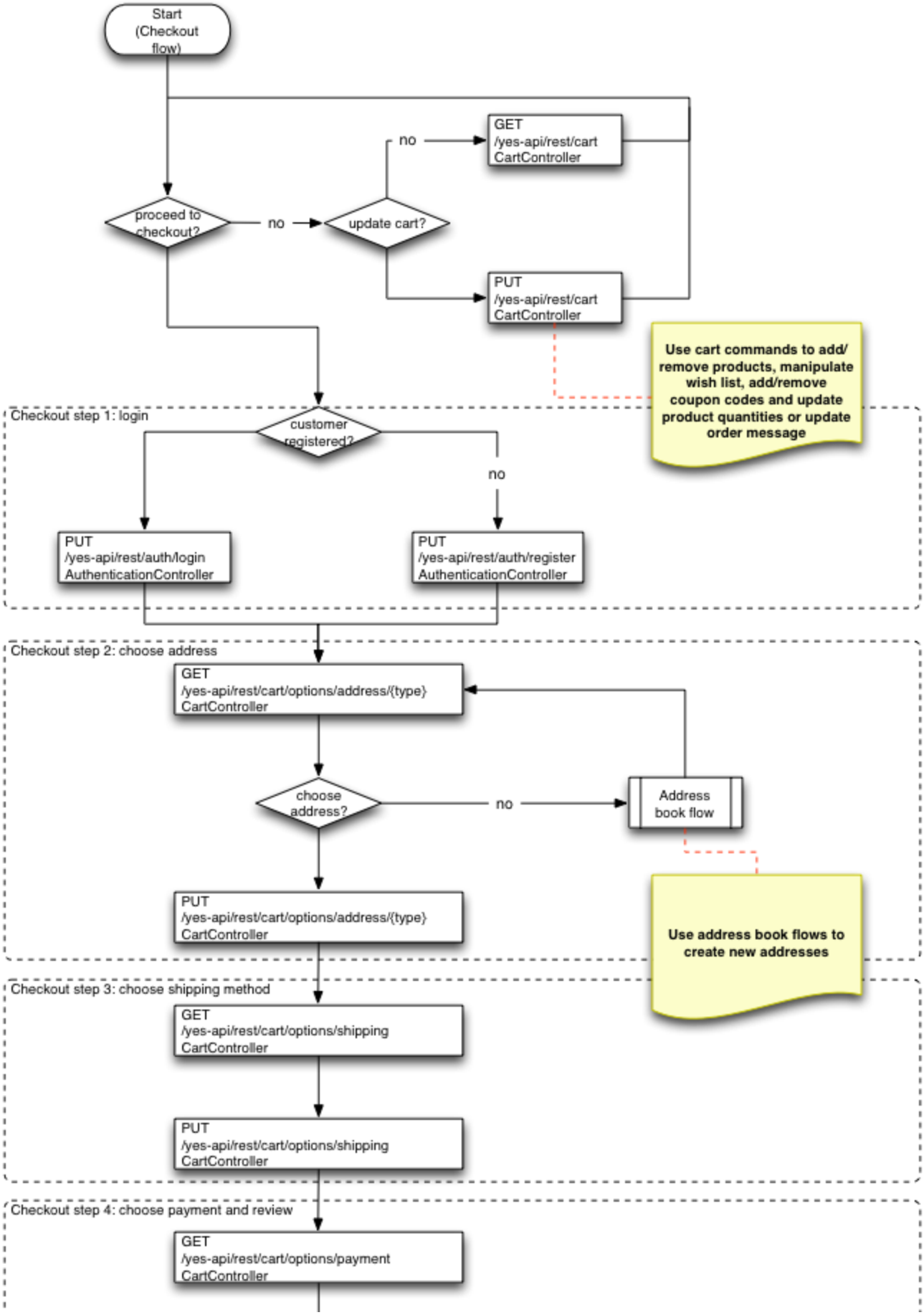
Checkout step 3: choose shipping method where customer selects the shipping method they wish to use. Note that shipping method influences available payment options.

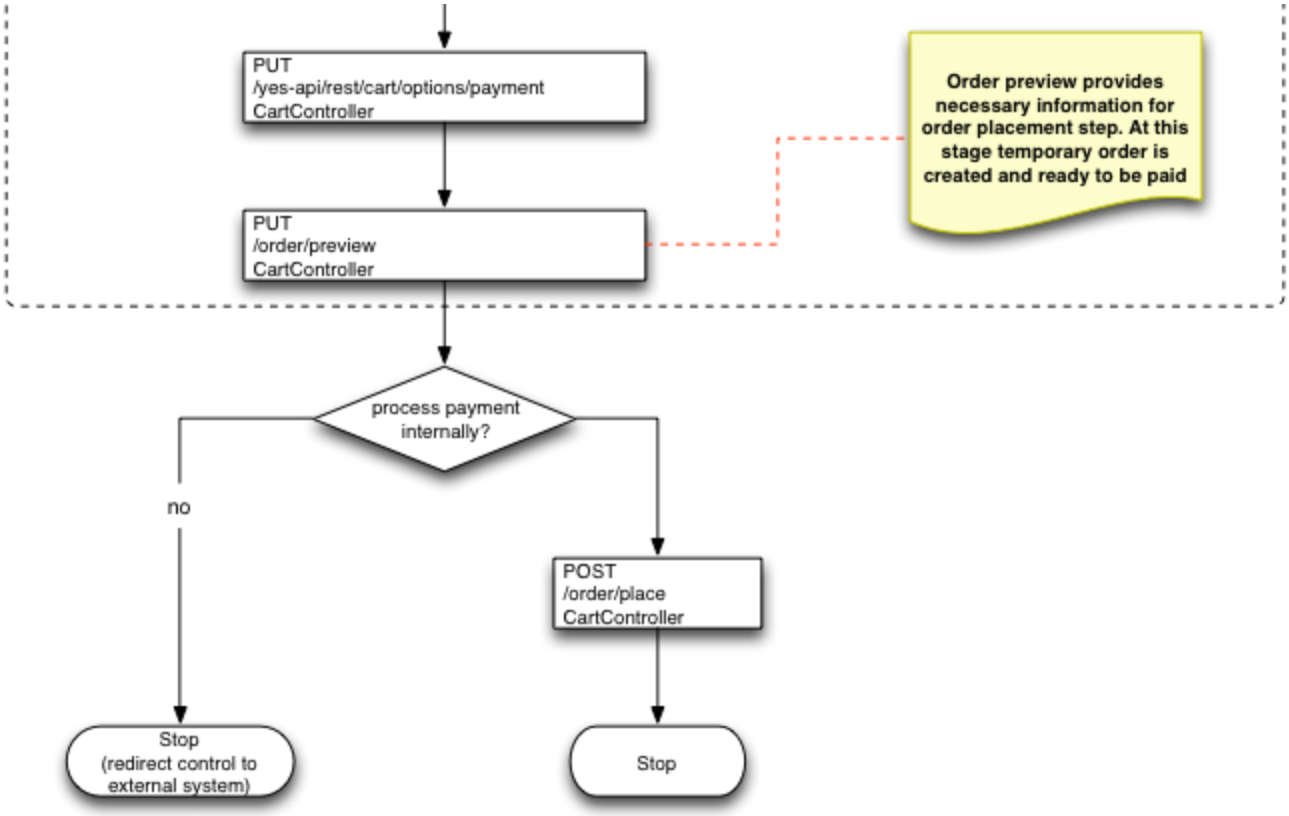
Checkout step 4: choose payment and review where customer selects the payment method they wish to use.

Step 4 finishes with **PUT /order/preview** call that shows all details for newly created temporary order and additional hints for payment processing. At this point third party system (e.g. mobile application) must identify whether this payment method requires transfer of control to third party payment system (e.g. redirect to PayPal web site, or some payment mobile app) or whether the platform can process this payment.

If the platform can process the payment it is made by **POST /order/place** call.

Note that payments that require payment confirmation callbacks will be processed by payment callback filters as they normally would for web site.


Start (Checkout flow)
proceed to checkout?
no
update cart?
no
GET /yes-api/rest/cart CartController
PUT /yes-api/rest/cart CartController
Use cart commands to add/ remove products, manipulate wish list, add/remove coupon codes and update product quantities or update order message
Checkout step 1: login
customer registered?
no
PUT /yes-api/rest/auth/login AuthenticationController
PUT /yes-api/rest/auth/register AuthenticationController
Checkout step 2: choose address
GET /yes-api/rest/cart/options/address/{type} CartController
choose address?
no
Address book flow
PUT /yes-api/rest/cart/options/address/{type} CartController
Use address book flows to create new addresses
Checkout step 3: choose shipping method
GET /yes-api/rest/cart/options/shipping CartController
PUT /yes-api/rest/cart/options/shipping CartController
Checkout step 4: choose payment and review
GET /yes-api/rest/cart/options/payment CartController



Resources

- REST API overview
- Live Swagger UI (Commerce API)
- Live Swagger UI (Mission Control API) ⚠️ if "token expired" message appears, please refresh the page

> More in-depth articles on shop specific REST API configurations are available upon request