REST API

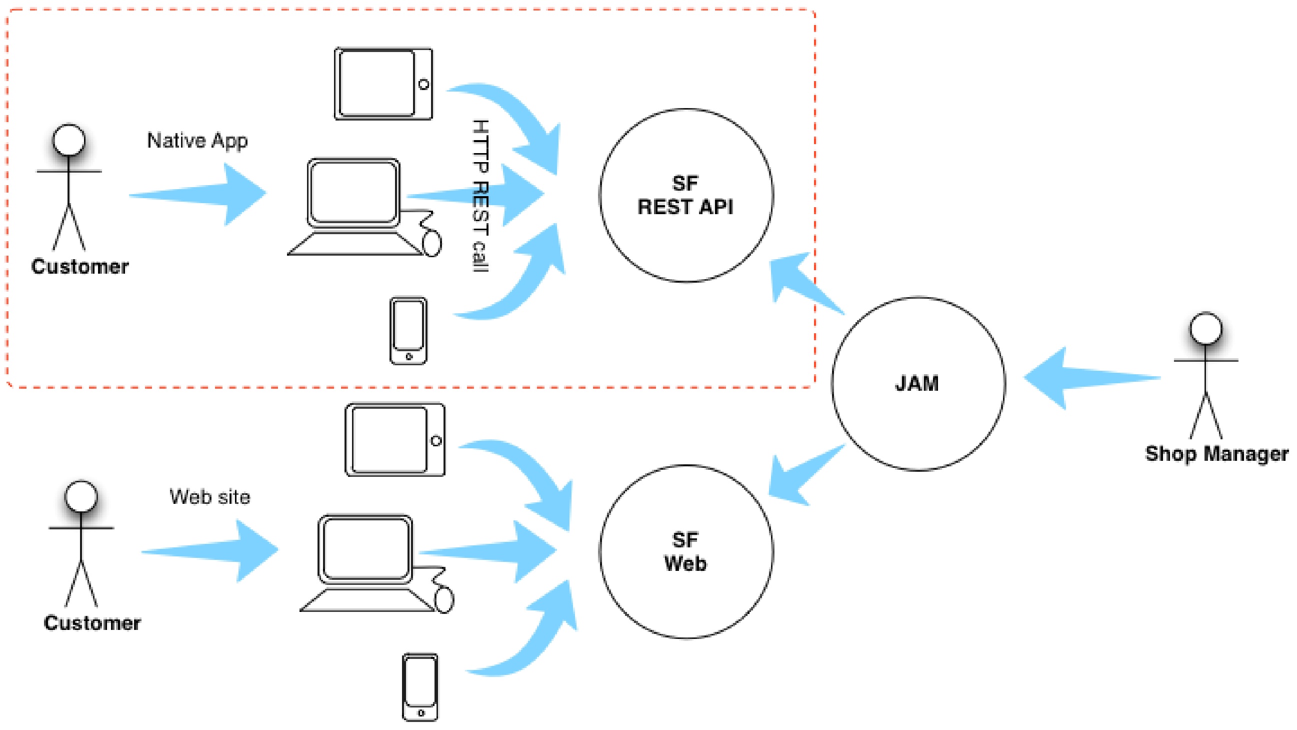
Overview

REST API is a web application which has **all functionality of the storefront** but lacks the customer facing parts. In essence this is the **facade of services that provide the data and drive storefront functions** such as search and navigation, content, price and promotion calculation, authentication, checkout process and customer profile amendment etc.

The rationale for having such web application is twofold: native applications and external CMS.

Apps

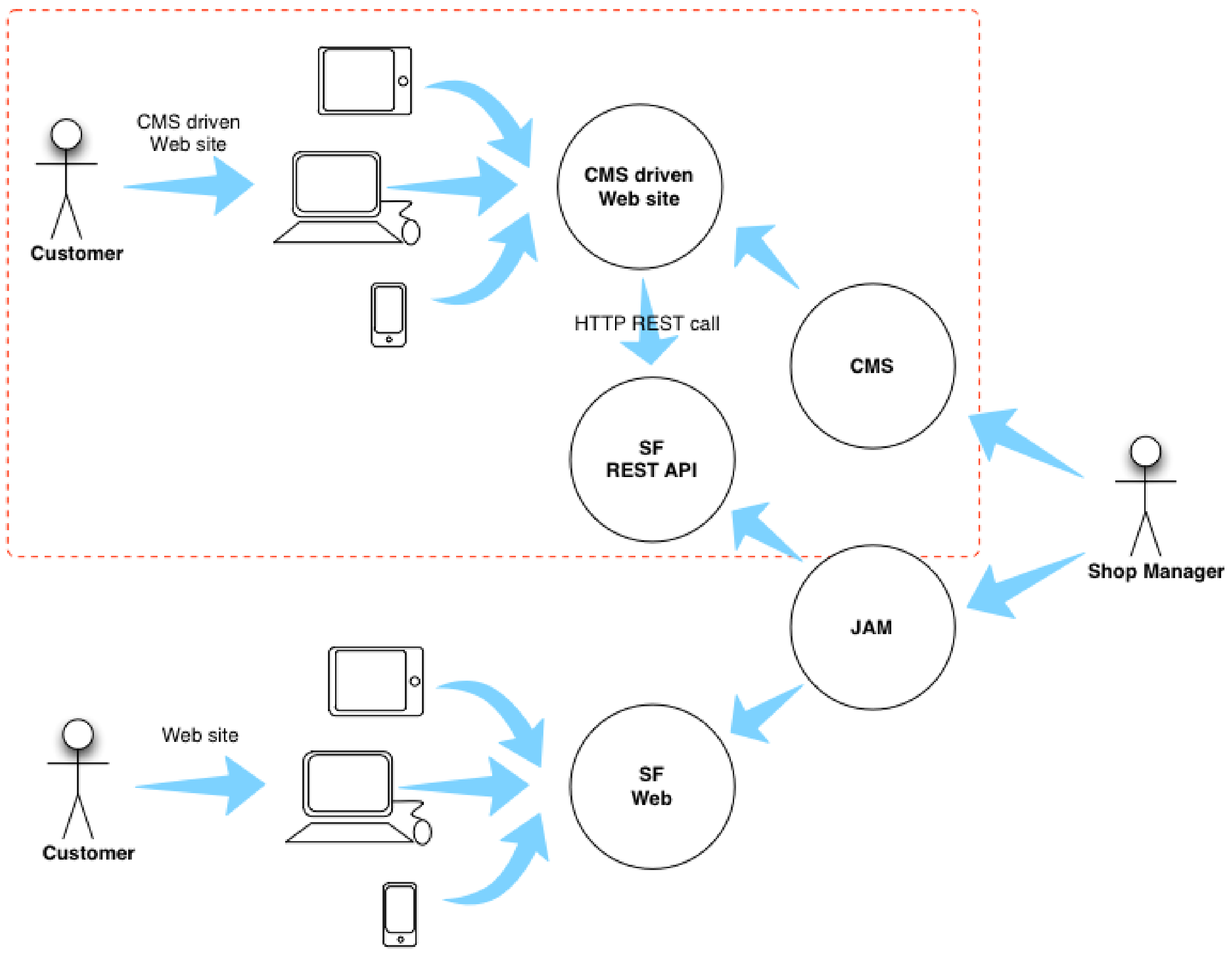
REST API can be used by mobile, tablets and desktop applications to support clients that wish to **develop native applications that use the platform as an e-commerce engine**. This is especially true in recent years with booming iPhone and Android **mobile applications**. However it is not limited just to mobile as there is **potential for specialised call centre and business-to-business (B2B) desktop applications**.



External CMS

With rise of **specialised CMS software** many businesses wish to manage their Internet presence using unified approach via tools such as Adobe

CMS and Alfresco (amongst many others). **REST API allow to call e-commerce services from within those external CMS** supplying catalog, profile and order data as well as facilitating storefront functions. Therefore corporate branding is maintained by external CMS but all data presented to client and complex e-commerce service layer is provided by the platform.



Technical specification

REST API provides a number of interfaces that support **JSON** and **XML** formats. Both request and response body can be either JSON or XML depending on the **Accept** and **Content-type** headers of any particular request.

Authentication is managed via token which is supplied as a cookie and a header via authentication REST API calls.

For detailed documentation on interfaces provided by REST API please refer to swagger spec

Platform 3.7.0+ versions take a SaaS API approach whereby sales channel is specified using **X-SALES-CHANNEL** header thus removing the need for having specific domain names.

Each API request has an optional authentication token to correlate with customer virtual session. In versions up to 3.6.x it was accomplished using **yc** header, in 3.7.0+ this header is renamed to **X-CW-TOKEN**.

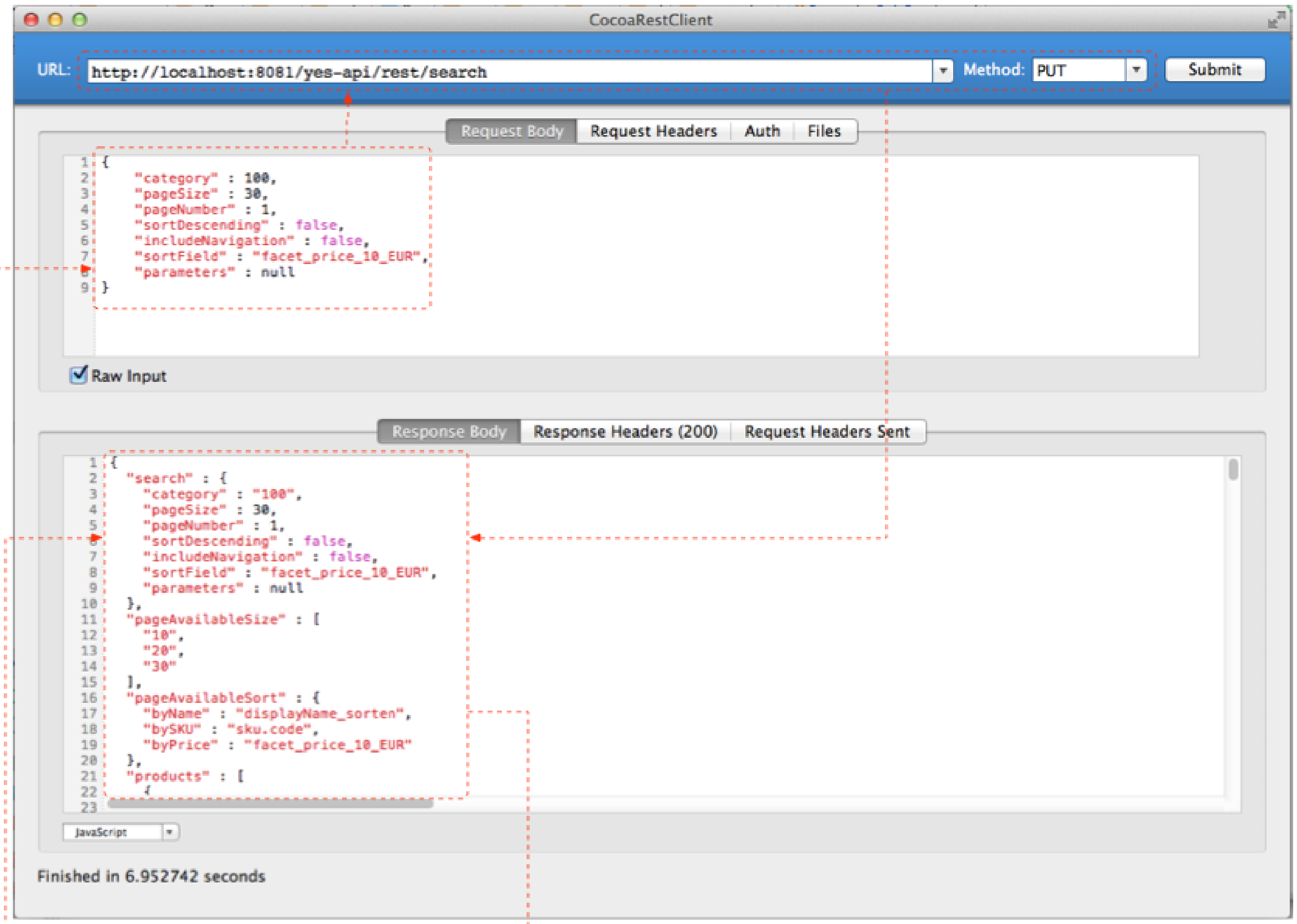
CORS support has been introduced in 3.7.0+ and is achieved through sales channel configuration attribute **SHOP_CORS_ALLOWED_ORIGINS** which is a CSV of Allowed origins verified against the **Origin** header in the CORS request.

Hands on

We recommend reviewing REST API basics cookbook to get more insight on using these interfaces, however please refer to the latest swagger spec for the most up to date API definitions and capabilities.

Figure 1 and 2 below show example of search JSON and XML responses. Note that **it is possible to mix and match body content type** as demonstrated in figure 2 with request made as JSON but response received as XML.

Figure 1: Example search call using JSON mode



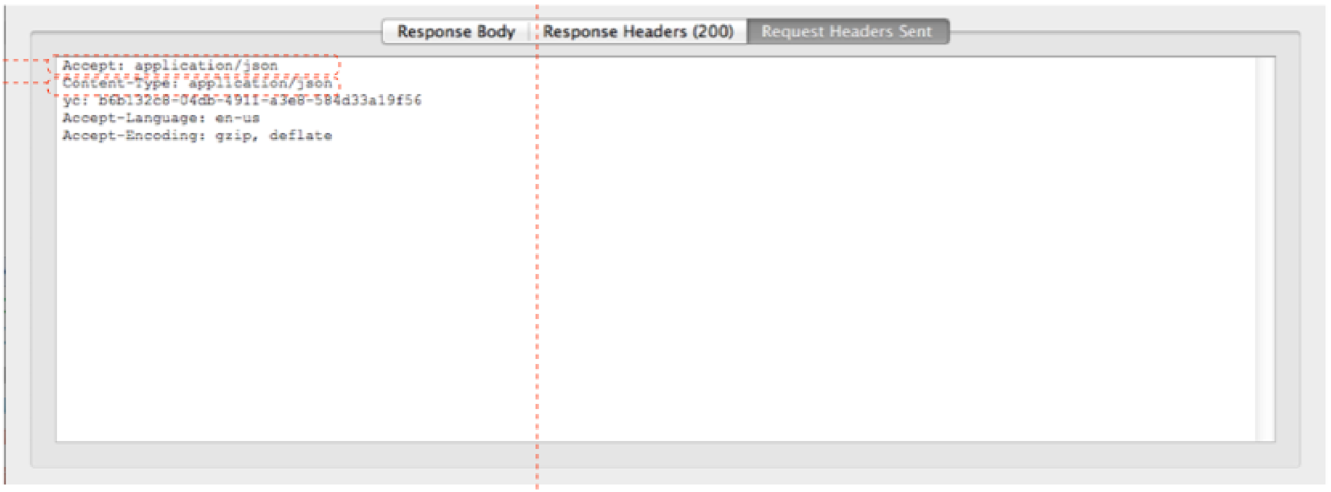
URL: http://localhost:8081/yes-api/rest/search

Method: PUT

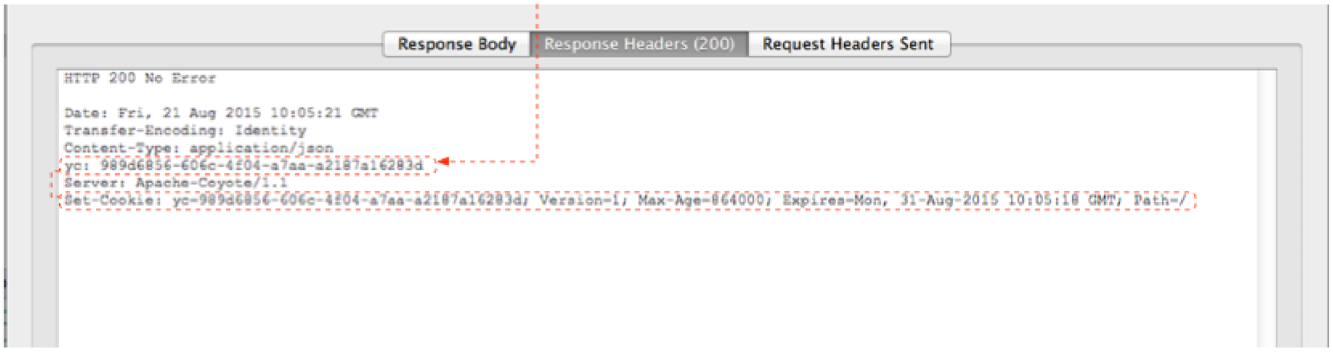
```
{
    "category" : 100,
    "pageSize" : 30,
    "pageNumber" : 1,
    "sortDescending" : false,
    "includeNavigation" : false,
    "sortField" : "facet_price_10_EUR",
    "parameters" : null
}
```

```
{
  "search" : {
    "category" : "100",
    "pageSize" : 30,
    "pageNumber" : 1,
    "sortDescending" : false,
    "includeNavigation" : false,
    "sortField" : "facet_price_10_EUR",
    "parameters" : null
  },
  "pageAvailableSize" : [
    "10",
    "20",
    "30"
  ],
  "pageAvailableSort" : {
    "byName" : "displayName_sorten",
    "bySKU" : "sku.code",
    "byPrice" : "facet_price_10_EUR"
  },
  "products" : [
    {
```

Finished in 6.952742 seconds



```
Accept: application/json
Content-Type: application/json
yc: b6b132c8-04db-4911-a3e8-584d33a19f56
Accept-Language: en-us
Accept-Encoding: gzip, deflate
```



```
HTTP 200 No Error

Date: Fri, 21 Aug 2015 10:05:21 GMT
Transfer-Encoding: Identity
Content-Type: application/json
yc: 989d6856-606c-4f04-a7aa-a2187a16283d
Server: Apache-Coyote/1.1
Set-Cookie: yc=989d6856-606c-4f04-a7aa-a2187a16283d; Version=1; Max-Age=864000; Expires=Mon, 31-Aug-2015 10:05:18 GMT; Path=/
```

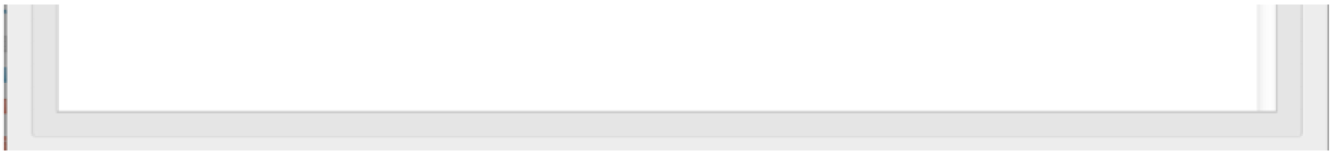
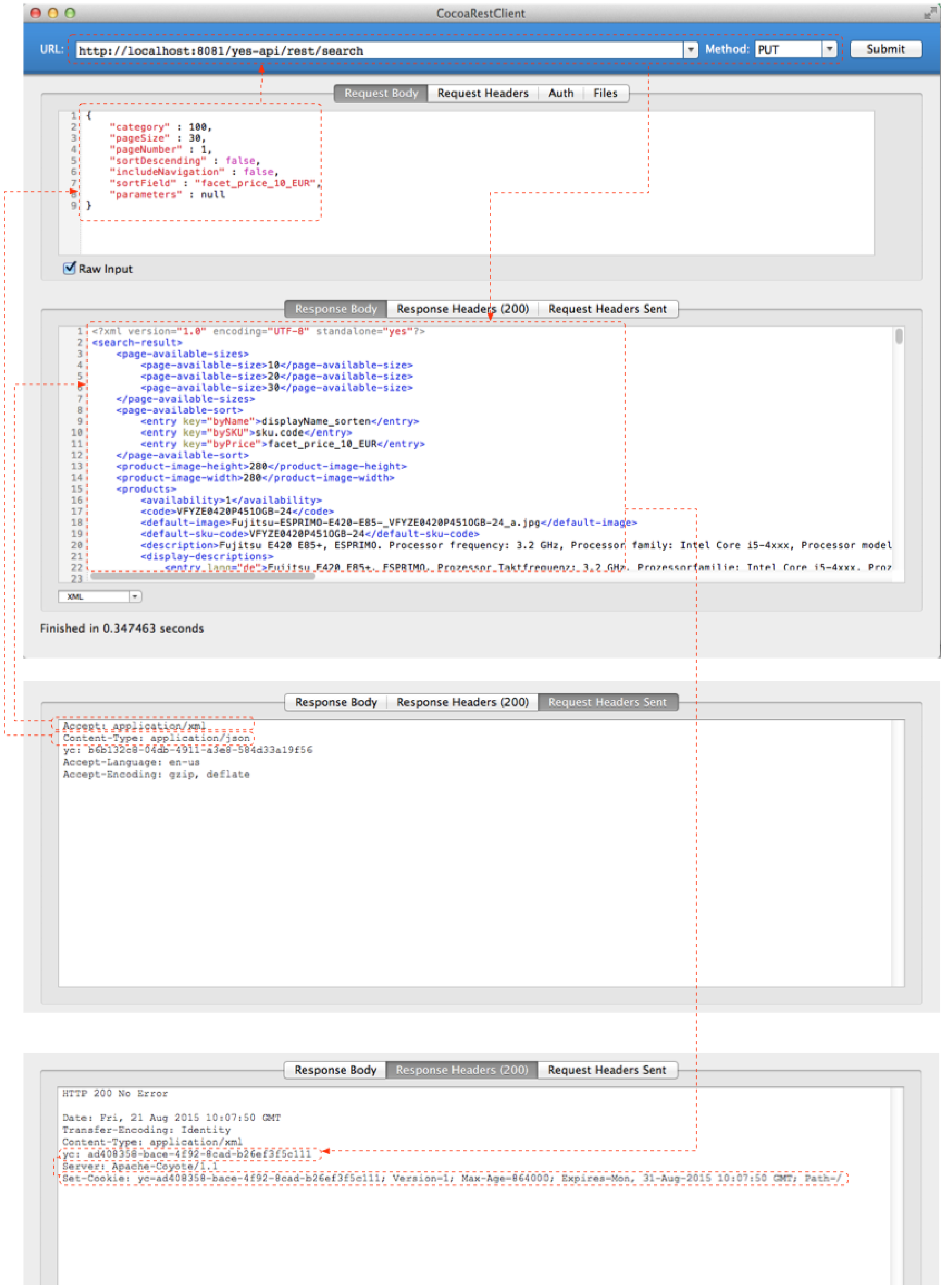


Figure 2: Example search call using JSON request mode and XML response mode



```
http://localhost:8081/yes-api/rest/search
```

Method: PUT

```
{
    "category" : 100,
    "pageSize" : 30,
    "pageNumber" : 1,
    "sortDescending" : false,
    "includeNavigation" : false,
    "sortField" : "facet_price_10_EUR",
    "parameters" : null
}
```

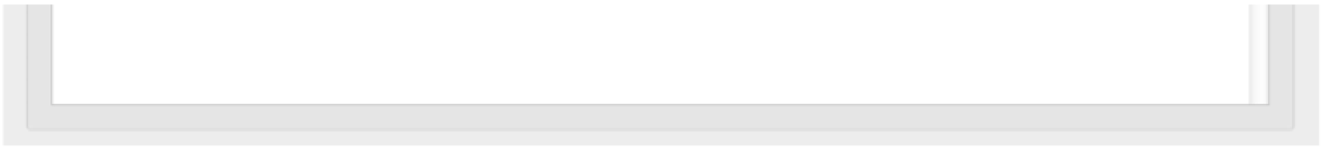
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<search-result>
    <page-available-sizes>
        <page-available-size>10</page-available-size>
        <page-available-size>20</page-available-size>
        <page-available-size>30</page-available-size>
    </page-available-sizes>
    <page-available-sort>
        <entry key="byName">displayName_sorten</entry>
        <entry key="bySKU">sku.code</entry>
        <entry key="byPrice">facet_price_10_EUR</entry>
    </page-available-sort>
    <product-image-height>280</product-image-height>
    <product-image-width>280</product-image-width>
    <products>
        <availability>1</availability>
        <code>VFYZE0420P4510GB-24</code>
        <default-image>Fujitsu-ESPRIMO-E420-E85-_VFYZE0420P4510GB-24_a.jpg</default-image>
        <default-sku-code>VFYZE0420P4510GB-24</default-sku-code>
        <description>Fujitsu E420 E85+, ESPRIMO. Processor frequency: 3.2 GHz, Processor family: Intel Core i5-4xxx, Processor model
        <display-descriptions>
            <entry lang="de">Fujitsu E420 E85+, ESPRIMO. Prozessor Taktfrequenz: 3.2 GHz, Prozessorfamilie: Intel Core i5-4xxx, Proz
```

Finished in 0.347463 seconds

```
Accept: application/xml
Content-Type: application/json
yc: b6b132c8-04db-4911-a3e8-584d33a19f56
Accept-Language: en-us
Accept-Encoding: gzip, deflate
```

```
HTTP 200 No Error

Date: Fri, 21 Aug 2015 10:07:50 GMT
Transfer-Encoding: Identity
Content-Type: application/xml
yc: ad408358-bace-4f92-8cad-b26ef3f5c111
Server: Apache-Coyote/1.1
Set-Cookie: yc=ad408358-bace-4f92-8cad-b26ef3f5c111; Version=1; Max-Age=864000; Expires=Mon, 31-Aug-2015 10:07:50 GMT; Path=/
```



CORS 3.7.0+

In order to enable CORS go to shop attributes, find **SHOP_CORS_ALLOWED_ORIGINS** configuration and define CSV list of allowed origins that are required.

Do not add leading or trailing spaces in this CSV.

Correct: "https://demo.yes-cart.org,https://localhost:8081,https://localhost:5555"

Incorrect: " https://demo.yes-cart.org,https://localhost:8081 , https://localhost:5555"

If you are setting up your own

You can test your CORS configuration using cURL, for example:

Pre-flight request check

```
# Options request which lists methods and headers for retrieving cart
curl -v -X OPTIONS "http://localhost:8081/api/rest/cart" -H "Origin:
http://localhost" -H "Access-Control-Request-Method: POST" -H
"Access-Control-Request-Headers: X-SALES-CHANNEL, X-CW-TOKEN, accept"
# typical response
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8081 (#0)
> OPTIONS /api/rest/cart HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.63.0
> Accept: */*
> Origin: http://localhost
> Access-Control-Request-Method: POST
> Access-Control-Request-Headers: X-SALES-CHANNEL, X-CW-TOKEN, accept
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Access-Control-Allow-Origin: http://localhost
< Vary: Origin
< Access-Control-Allow-Methods: GET,POST,PUT,DELETE,OPTIONS,HEAD
< Access-Control-Allow-Headers: X-SALES-CHANNEL, accept
< Access-Control-Allow-Credentials: true
< Content-Length: 0
< Date: Sun, 06 Sep 2020 10:06:31 GMT
<
```

Sending the request will require appropriate headers to be sent:

Valid request

```
# GET request to retrieve the cart from an allowed origin
curl -v -X GET "http://localhost:8081/api/rest/cart" -H "accept:
application/xml" -H "X-SALES-CHANNEL: localhost" -H "Origin:
http://localhost"
# Typical positive response
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8081 (#0)
> GET /api/rest/cart HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.63.0
> accept: application/xml
> X-SALES-CHANNEL: localhost
> Origin: http://localhost
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Access-Control-Allow-Origin: http://localhost
< Vary: Origin
< Access-Control-Allow-Credentials: true
< Pragma: no-cache
< Expires: Thu, 01 Jan 1970 00:00:00 GMT
< Cache-Control: no-cache
< Cache-Control: no-store
< Set-Cookie: X-CW-TOKEN=304b592c-3963-4dcc-9bbf-c8b5973d0a3a; Version=1;
Max-Age=864000; Expires=Wed, 16-Sep-2020 10:09:01 GMT; Path=/
< X-CW-TOKEN: 304b592c-3963-4dcc-9bbf-c8b5973d0a3a
< Content-Type: application/xml
< Transfer-Encoding: chunked
< Date: Sun, 06 Sep 2020 10:09:02 GMT
<
* Connection #0 to host localhost left intact
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> ...
```

Invalid request

```
# GET request to retrieve the cart from an unknown origin
curl -v -X GET "http://localhost:8081/api/rest/cart" -H "accept:
application/xml" -H "X-SALES-CHANNEL: localhost" -H "Origin:
http://nonallowed.com"
# Typical negative response
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8081 (#0)
> GET /api/rest/cart HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.63.0
> accept: application/xml
> X-SALES-CHANNEL: localhost
> Origin: http://nonallowed.com
>
< HTTP/1.1 403 Forbidden
< Server: Apache-Coyote/1.1
< Content-Length: 20
< Date: Sun, 06 Sep 2020 10:15:07 GMT
<
* Connection #0 to host localhost left intact
Invalid CORS request
```