## **Overview**

Most OrderCloud.io Resources include a method for listing items from that Resource. The items from such a call could very well be in the thousands so being able to narrow down the range of the items returned is critical, this can be done with the use of filters.

## **Filtering On XP**

Let's examine a common scenario of filtering on an extended property (XP) field. Suppose our XP field looks like this:

{  
"xp": {  
 "MoreInfo": {  
 "TeamName": "Example Team Name",  
 "Gender": "Male or Female"  
 }  
 }  
}

We can use dot notation to access deeply nested values. The following call will return all buyers with team name "Tigers"

GET https://api.ordercloud.io/buyers/xyz/users?xp.MoreInfo.TeamName=Tigers HTTP/1.1

## **Fuzzy Searches**

Fuzzy matches are supported using the \* wildcard character.

GET https://api.ordercloud.io/buyers/xyz/users?LastName=\*Smith&FirstName=John\* HTTP/1.1

This will return both "John Smith" and "Johnny McSmooth".

## **Logical OR**

You can also use | as a logical OR.

GET https://api.ordercloud.io/buyers/xyz/users?LastName=Smith|Jo\*&FirstName=Johnny HTTP/1.1

This will return "John Smith", "Johnny Jones", and "John Johnson". Maybe you want "John Smith" but not "John Jones".

## **Negate and Logical AND**

You can negate your conditions by prefixing them with !, and logically AND them together by simply providing the same parameter multiple times.

GET https://api.ordercloud.io/buyers/xyz/users?LastName=!Smith&LastName=!Jones HTTP/1.1

This will return all users *except* those with last name "Smith" or "Jones".

## **Comparison Operators**

Dates and numeric values support > (greater than) and < (less than) prefixes.

GET https://api.ordercloud.io/buyers/xyz/users?DateCreated=>2015-01-01 HTTP/1.1

For a more advanced example, let's say you want users whose ID is the range of 0 to 9 inclusive. Ranges are not directly supported, but you can use the existing features to achieve this.

GET https://api.ordercloud.io/buyers/xyz/users?ID=0|1|2|3|4|5|6|7|8|9 HTTP/1.1

Of course that's not going to be feasible if the range is very large or you're dealing with floating-point numbers. A better way would be to leverage > and <, but we want the range to be inclusive, and there are no >= or <= operators. We can however leverage the ! operator.

Here we're saying "give me all users whose ID is not less than 0 and not greater than 9", which is effectively equivalent to our 0-9 range.

GET https://api.ordercloud.io/buyers/xyz/users?ID=!<0&ID=!>9 HTTP/1.1

## **Performance Considerations**

Your data is highly indexed for fast retrieval using any of the methods above, including deep XP object graphs. However, there are a few things to keep in mind to keep search and listing functionality efficient. Filters with many OR conditions or with values that start with the \* wildcard character may be particularly performance-sensitive. The larger the set of data, the more likely you are to notice any performance impact. It pays to know the data model well, and use grouping constructs like User Groups and Categories effectively.

## **Combining Queries**

Filtering can be mixed and matched with Searching and Sorting to give you ultimate control when defining what is returned from a list.