## **## Overview**

The reality of all enterprise platforms is the need to customize the data model. To accommodate this, OrderCloud.io designed a property called Extended Properties (XP) to allow enterprise customization of the OrderCloud data model. OrderCloud.io exposes the `{ xp: {} }` property on most resources allowing you to apply JSON objects. The JSON object can be as complex and deeply-nested as necessary. Additionally, the XP property is available for filtering, sorting and searching in all list endpoints.

Extended Properties allows you to overcome platform rigidity. So, to optimize the use of our data model, and to help you fully implement your B2B scenarios, we created a schema-less solution with XP and exposed it on virtually every API resource. We may not have `Product.YourSpecialDataPoint`, but we do have `Product.xp.YourSpecialDataPoint`.

## **## Add, Update and Remove XP**

The entire XP object must be <strong>8000 bytes</strong> or less and must be a valid JSON object. Any key-value pairs of numbers, strings, booleans, arrays, and even other objects can be used.

## **## Adding XP**

Let's say one of the requirements for your solution includes storing the age and gender of users. You can accomplish this by storing those data points in the user's XP. If you do not include JSON, XP will be set to null by default on all parent objects. To declare XP, simply replace the null value with the valid JSON.

```

POST https://api.ordercloud.io/buyers/newbuyer/users HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json; charset=UTF-8  
  
{  
 "Username": "janesmith",  
 "Password": "test12345",  
 "FirstName": "Jane",  
 "LastName": "Smith",  
 "Email": "jsmith@company.com",  
 "Phone": "555-555-5555",  
 "TermsAccepted": null,  
 "Active": true,  
 "xp": {  
 "Gender": "Female",  
 "Age" : 26  
 },  
 "SecurityProfileID": "FullAccess"  
}

```

## **## Nesting XP**

Let's say requirements have shifted and the solution now requires the ability to store information about the user's employment details, specifically job title and department. This can easily be accomplished using nested objects within XP.

```

PUT https://api.ordercloud.io/buyers/newbuyer/users/userID HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json; charset=UTF-8  
  
{  
 "Username": "janesmith",  
 "Password": "test12345",  
 "FirstName": "Jane",  
 "LastName": "Smith",  
 "Email": "jsmith@company.com",  
 "Phone": "555-555-5555",  
 "TermsAccepted": null,  
 "Active": true,  
 "xp": {  
 "Gender": "Female",  
 "Age" : 26,  
 "EmploymentDetails" : {  
 "Position": "Developer",  
 "Department": "Tech"  
 }  
  
 },  
 "SecurityProfileID": "FullAccess"  
}

```

## **## Modifying XP**

Now let's say Jane Smith receives a promotion. To update her job title we can use `PATCH` to modify the relevant data. Instead of sending the entire user object, we can send the XP key and the object that we want to update, the other XP (gender and age) will persist.

```

PUT https://api.ordercloud.io/buyers/newbuyer/users/userID HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json; charset=UTF-8  
  
{  
 "xp": {  
 "EmploymentDetails" : {  
 "Position": "Senior Developer",  
 "Department": "Tech"  
 }  
 }  
}

```

## **## Deleting XP**

The only way to remove a specific xp from your resource is to use the Update (PUT) method. First GET the resource, then copy the response body returned, paste it into your PUT request and omit the xp key/value you wish to remove. Alternatively, if you don’t need it actually deleted, you are able to set any xp’s value to null.```

GET [https://api.ordercloud.io/buyers/{buyerid}/users/{userid](https://api.ordercloud.io/buyers/%7Bbuyerid%7D/users/%7Buserid)} HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json; charset=UTF-8

Response:

{

"ID": "",

"Username": "",

"FirstName": "",

"LastName": "",

"Email": "",

"Phone": "",

"TermsAccepted": "0001-01-01T00:00:00+00:00",

"Active": false,

"xp": {

“Test”: true,

“DOB”: “01/01/2018”

},

"AvailableRoles": [

""

]

}

```

Then:

```

UPDATEhttps://api.ordercloud.io/buyers/newbuyer/users/userID HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json; charset=UTF-8  
{

"ID": "",

"Username": "",

"FirstName": "",

"LastName": "",

"Email": "",

"Phone": "",

"TermsAccepted": "0001-01-01T00:00:00+00:00",

"Active": false,

"xp": {

“DOB”: “01/01/2018”

}

"AvailableRoles": [

""

]

}

```

## **## Searching on XP**

Not only can XP be used to extend the functionality of your application, but you can use filters to search for indexed xp values on any given resource. After all, how much benefit would XP provide if you couldn't query a subset of objects based on a specific XP value? All of the filtering capabilities that apply to regular values on OrderCloud.io objects also apply to XP values. This means you can search with all of the standard operators available (`=`,`<`,`>`). Below is an example of filtering for a deeply nested value in XP:

```

GET https://api.ordercloud.io/buyers/newbuyer/users?xp.EmploymentDetails.Department=Tech HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json; charset=UTF-8

```

<div class="note">Note how the dot (".") notation is used in the query string to filter on the specified deeply nested XP field. Check out the searching, filtering and sorting guides to learn more about querying.</div>

## **## Summary**

XP is a very powerful feature to extend the capabilities of your application. It provides the flexibility developers need to meet challenging requirements, giving you the ability to provide a highly customized solutions. Head over to the API Console to try it for yourself!