## **## Introduction**

A very simplistic definition of a catalog is a collection of products visible to an individual user. While concise, this definition does not lend itself well to more complex catalogs which require customization and flexibility, two integral tenets of OrderCloud.io. So, defining a catalog by breaking it up into individual resources allows us to understand how to create the customization and flexibility we need.

Instead of having a single price, we have what is called a Price Schedule. You can think of this as a framework for how pricing should be determined rather than a static price. This framework has several variables: quantity (think price breaks), specs (options for color, size, etc), and perspective (who is buying the product) which can all impact the product's pricing. Each resource is part of a puzzle that can be defined independently, but it is not until everything is assembled (in our case, assigned) that a snapshot of pricing becomes available.

The last *optional* relationship is between products and categories. These relationships enable an admin user to control the browsing experience of the buyer catalog.

This flexibility allows you to fulfill virtually every possible catalog requirement in the B2B landscape. Returning to our puzzle analogy you can have a completely different snapshot the buyer catalog by altering one of these pieces or their relationships to one another.

## **## Prerequisites**

Before we begin making API calls, you will need an organization and at least one User <strong>[Security Profile](https://documentation.ordercloud.io/platform-guides/getting-started/using-the-dashboard#SecurityProfiles)</strong>. This profile will dictate which endpoints your buying experience can use so be sure it at the very least has `BuyerReader`. If you've followed along with our [Quick Start Guide](https://documentation.ordercloud.io/platform-guides/getting-started/quick-start-guide), you should already have one organization set up with an application and an `access\_token` ready to go.

## **## 1. Create a Buyer Organization**

Create the buyer organization, which will contain the assignment relationships of our catalog:

<div class="api-reference">API Reference: [Create New Buyer](<https://documentation.ordercloud.io/api-reference#Buyers_Create>)</div>

```

POST https://api.ordercloud.io/v1/buyers HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json; charset=UTF-8  
  
{  
 "ID": "newbuyer",  
 "Name": "New Buyer",  
 "Active": true,  
 "xp": null  
}

```

In order to keep things simple, let's assign a security profile to the buyer we just created. Buyer users will have the same level of access due to the OrderCloud.io inheritance configurations.

<div class="api-reference">API Reference: [Save Assignment](<https://documentation.ordercloud.io/api-reference#SecurityProfiles_SaveAssignment>)</div>

```

POST https://api.ordercloud.io/v1/securityprofiles/assignments HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json; charset=UTF-8  
  
{  
 "SecurityProfileID": "ID\_OF\_YOUR\_USER\_SECURITY\_PROFILE",  
 "BuyerID": "newbuyer"  
}

```

## **## 2. Create a Buyer User**

Next, Create a Buyer User. This is the user that will ultimately be able to view the product catalog:

<div class="api-reference">API Reference: [Create New Buyer](<https://documentation.ordercloud.io/api-reference#Users_Create>)</div>

```

POST https://api.ordercloud.io/v1/buyers/{buyerID}/users HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json; charset=UTF-8  
  
{  
 "ID": "1",  
 "Username": "user01",  
 "Password": "password",  
 "FirstName": "First",  
 "LastName": "User",  
 "Email": "user01@example.com",  
 "Phone": "555-555-5555",  
 "TermsAccepted": null,  
 "Active": true,  
 "xp": null  
}

```

## **## 3. Create a User Group**

User groups are assortments of users with similar purchasing and viewing experiences. Assignments to the user group will be inherited by members of that group. This greatly reduces the number of potential assignment calls. For example, imagine a user group with 100 members. Rather than making 100 separate assignments you can make one assignment to the user group which will affect all users within that group.

For this simple catalog we only have one user group, therefore a single buyer user experience. While it is possible to make assignments directly to individual users, we recommend keeping your assignments at the user group level for better performance. You can then differentiate the buying experience by introducing additional groups.

<div class="api-reference">API Reference: [Create New User Group](<https://documentation.ordercloud.io/api-reference#UserGroups_Create>)</div>

```

POST https://api.ordercloud.io/v1/buyers/{buyerID}/usergroups HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json; charset=UTF-8  
  
{  
 "ID": "A",  
 "Name": "UserGroup A",  
 "Description": "My First UserGroup",  
 "xp": null  
}

```

## **## 4. Assign the User to the User Group**

Create a relationship between the buyer user we created in Step 2 and the new User Group:

<div class="api-reference">API Reference: [Save User Assignment](https://documentation.ordercloud.io/api-reference#UserGroups\_SaveUserAssignment)</div>

```

POST https://api.ordercloud.io/v1/buyers/{buyerID}/usergroups/assignments HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json; charset=UTF-8  
  
{  
 "UserGroupID": "A",  
 "UserID": "1"  
}

```

## **## 5. Create a Product**

Create at least one product to be a part of the buyer catalog. We only providing minimal product information to keep things fast and simple:

<div class="api-reference">API Reference: [Create New Product](<https://documentation.ordercloud.io/api-reference#Products_Create>)</div>

```

POST <https://api.ordercloud.io/v1/products> HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json  
  
{  
 "ID": "X",  
 "Name": "Product X",  
 "Description": "",  
 "QuantityMultiplier": 1,  
 "Active": true  
}

```

## **## 6. Create a Price Schedule**

A price schedule can be thought of as a framework for how pricing for a product will be determined rather than a static price (although pricing can be static if defined as such). There are many nuanced features hidden within the price schedule model that will be covered in future guide; so again, we're only providing boilerplate information here to keep things simple:

<div class="api-reference">API Reference: [Create New Price Schedule](<https://documentation.ordercloud.io/api-reference#PriceSchedules_Create>)</div>

```

POST <https://api.ordercloud.io/v1/priceschedules> HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json  
  
{  
 "ID": "PS\_X",  
 "Name": "Price Schedule X",  
 "MaxQuantity": null,  
 "OrderType": "Standard",  
 "PriceBreaks": [  
 {  
 "Quantity": 1,  
 "Price": 10.0  
 }  
 ],  
 "xp": null  
}

```

The value that we care most about is `PriceBreaks`. This value is an array of objects with `Quantity` and `Price` inside the object. This should be interpreted as: "Each unit costs $10 for all total quantities." If we had another with Quantity set to 10 and Price set to 8, that would mean, "Each unit costs $10 for any quantity between 1 and 10 units, and $8 thereafter."

## **## 7. Assign the Product to the Price Schedule & User Group**

This is the most unique type of assignment in OrderCloud.io, as it ties \***three\*** entities together opposed to the usual \***two\***. This means that you can assign a single product to two user groups with different pricing for each of them, otherwise known as \***individual-based pricing\***.

Go ahead and assign product X to user group A using the price schedule PS\_X for the `PriceScheduleID`:

<div class="api-reference">API Reference: [Save Product Assignment](https://documentation.ordercloud.io/api-reference#Products\_SaveAssignment)</div>

```

POST <https://api.ordercloud.io/v1/products/assignments> HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json  
  
{  
 "ProductID": "X",  
 "ScheduleID": "PS\_X",  
 "UserGroupID": "A",  
 "BuyerID": "newbuyer"  
}

```

You may have noticed we included `BuyerID` in our request body. This is so OrderCloud.io knows where to look for the user and user group, it does not mean this product will be assigned to the entire buyer organization at this price point. In order to do that you would leave out `UserGroupID`, which would tell OrderCloud.io to assign product X at price PS\_X to the whole buyer organization.

## **## Hold up, wait a minute!**

It's worth pausing here to mention that after Step 7, we now have all relationships in place to view a product with pricing from a buyer user's perspective. To view this information, we [authenticate as the buyer user](https://documentation.ordercloud.io/platform-guides/authentication/oauth2-workflows) we created in Step 2 and list the products from the user's perspective:

<div class="api-reference">API Reference: [List Products](https://documentation.ordercloud.io/api-reference#MeProducts\_ListProducts)</div>

```

GET https://api.ordercloud.io/v1/me/products HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json

{  
 "Meta": {  
 "Page": 1,  
 "PageSize": 20,  
 "TotalCount": 1,  
 "TotalPages": 1,  
 "ItemRange": [1,1]  
 },  
 "Items": [  
 {  
  
 "PriceSchedule": {  
 "ID": "PS\_X",  
 "Name": "Price Schedule X",  
 "ApplyTax": false,  
 "ApplyShipping": false,  
 "MinQuantity": 1,  
 "MaxQuantity": null,  
 "UseCumulativeQuantity": false,  
 "RestrictedQuantity": false,  
 "OrderType": "Standard",  
 "PriceBreaks": [  
 {  
 "Quantity": 1,  
 "Price": 10.00  
 }  
 ],  
 "xp": null  
 }  
 "ID": "X",  
 "Name": "Product X",  
 "Description": "This is product X",  
 "QuantityMultiplier": 1,  
 "ShipWeight": null,  
 "Active": true,  
 "Type": "Static",  
 "InventoryEnabled": false,  
 "InventoryNotificationPoint": null,  
 "VariantLevelInventory": false,  
 "SpecCount": 0,  
 "xp": null,  
  
 "VariantCount": 0  
 }  
}

```

<div class="note">The previous 7 steps covered the complex relationships required for setting up *any* successful buyer catalog. The next few steps will cover the *optional* product to category relationship.<br>Be sure to switch back to your initial `access\_token` before continuing!</div>

**## 8. Create and Assign a Category**

Typical commerce catalogs organize their products using categories. To create this experience, we will need to create at least one category

<div class="api-reference">API Reference: [Save Assignment](https://devcenter.ordercloud.io/api-reference#Categories\_Create)</div>

```

POST [https://api.ordercloud.io/v1/catalogs/{catalogID}/categories](https://api.ordercloud.io/v1/catalogs/%7BcatalogID%7D/categories) HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json  
  
{  
 "ID": "CAT\_1",  
 "Name": "Category 1",  
 "Description": "This is category 1",  
 "xp": null,  
 "ListOrder": 1,  
 "Active": true  
}

```

<div class="note">`ListOrder` enables admin users to dictate the order of categories on your buyer application's catalog view.</div>

Like most assignments, categories can be assigned at the Buyer, Group, or User level. We'll assign this category to the entire buyer organization by leaving `UserGroupID` and `UserID` as null in our assignment object:

<div class="api-reference">API Reference: [Save Assignment](<https://devcenter.ordercloud.io/api-reference#Categories_SaveAssignment>)</div>

```

POST https://api.ordercloud.io/v1/catalogs/{catalogID}/categories/assignments HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json  
  
{  
 "CategoryID": "CAT\_1",  
 "UserID": null,  
 "UserGroupID": null  
}

```

## **## 9. Assign the Product to the Category**

Finally, assign `product X` to `"CategoryID": "CAT\_1"` with the following assignment request:

<div class="api-reference">API Reference: [Save Product Assignment](<https://devcenter.ordercloud.io/api-reference#Categories_SaveProductAssignment>)</div>

```

POST [https://api.ordercloud.io/v1/catalogs/{catalogID}/categories/productassignments](https://api.ordercloud.io/v1/catalogs/%7BcatalogID%7D/categories/productassignments) HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json  
  
{  
 "CategoryID": "CAT\_1",  
 "ProductID": "X",  
 "ListOrder": 1  
}

```

<div class="note">`ListOrder` enables admin users to control the order in which products are displayed within the category.</div>

Now that we've got our category set up you can use your **<strong>buyer user</strong>** access\_token to list out the categories the buyer user can see (in our case, this assignment is inherited from the buyer organization):

<div class="api-reference">API Reference: [List Categories](https://documentation.ordercloud.io/api-reference#MeCategories\_ListCategories)</div>

```

GET https://api.ordercloud.io/v1/me/categories HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json  
  
{  
 "Meta": {  
 "Page": 1,  
 "PageSize": 20,  
 "TotalCount": 1,  
 "TotalPages": 1,  
 "ItemRange": [1,1]  
 },  
 "Items": [  
 {  
 "CategoryID": "CAT\_1",  
 "UserID": null,  
 "UserGroupID": null  
 }  
 ]  
}

```

In a typical buyer application, once a category is selected you would list out the products under the category by passing in a categoryID param to the product list request:

```

GET <https://api.ordercloud.io/v1/me/products?categoryID=CAT_1> HTTP/1.1  
Authentication: Bearer put\_access\_token\_here  
Content-Type: application/json

```

## **## Conclusion**

You should now have a basic understanding of how to construct an OrderCloud.io buyer catalog that has individual-based pricing. You've taken the first steps towards truly understanding our API but there is still much more to learn! Continue reading our guides to gain a more detailed understanding of the platform.