## **Overview**

Our Angular SDK makes interacting with the OrderCloud.io API insanely easy. It provides one AngularJS factory called OrderCloud which can be used anywhere in your application. For the most part, the SDK is a 1:1 reflection of the OrderCloud.io API. Each resource in the API is a sub-service within OrderCloud. Getting the currently authenticated user works like so:

OrderCloud.Me.Get();

The Angular Seed comes with this SDK pre-installed and injected (as a bower package and angular dependency). If you're using the [seed project](http://localhost:3000/docs/angularjs/seed-project) and have followed the [Getting Started guide](http://localhost:3000/docs/angularjs/getting-started), you will not need to worry about installing and configuring the SDK for use. However, reading this overview will give you a deeper understanding of how your angular app works.

## **Installation**

If you're starting your own Angular project from scratch you can install the SDK via bower by running bower install ordercloud-ng-sdk --save from your command prompt. Once available, you will need to inject the module into your angular apps dependencies:

angular.module('myAngularApp', [orderCloud.sdk']);

## **Configuration**

Our Angular SDK requires some minimal information to communicate with a specific Admin or Buyer Organization. If you're using our AngularJS Seed, you will not need to define these angular constants as they will be automatically generated based on src/app/app.config.json when the application is built by Gulp. The Getting Started guide covers how to set up this file. However, if you're not using the seed you must add some constants to your application

angular.module('myAngularApp', ['orderCloud.sdk'])  
 .constant('clientid', 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx')  
 .constant('scope', 'FullAccess DevCenterImpersonate')  
 .constant('appname', 'My Angular App')  
 .constant('buyerid', 'xxxx')

.constant(‘catalogid’,’xxxx’)  
 .constant('authurl', 'https://auth.ordercloud.io/oauth/token')  
 .constant('apiurl', 'https://api.ordercloud.io')

clientid defines the API Client and can be retrieved from any organization or buyer API Client list in the dashboard.

scope is a space delimited list of claims the app will request from the Oauth server. Claims are unique for each API Client and can be limited further via each user's Security Profile.

appname is used for setting uniquely named cookies for access tokens and Buyer IDs.

buyerid is the ID of the Buyer to be loaded by default at runtime.

catalogid is the ID of the catalog that you will be interacting with. The catalog must be assigned the current buyer. The default catalog ID is the same as the buyer ID.

The values for authurl and apiurl are the base urls used by the SDK to talk to the OrderCloud.io API.

The Angular SDK has some additional sub-services built into it-- things that you will not find in the API Documentation. **These are perhaps the most important parts of the SDK to understand.** They are covered below.

## **Authentication**

Authentication is handled through the OrderCloud.Auth sub-service. You'll find a real world example in the Angular Seed's default [Login component](https://github.com/ordercloud-api/angular-seed/blob/master/src/app/login/login.js#L81). The sub-service has four methods for getting, storing, reading, and removing an OrderCloud.io access token to and from the user's cookies. If the phrase "access token" seems foreign to you, please read our post on Authentication Workflows.

Methods:

* GetToken - receives an object with user credentials (username, password) and return an access\_token when accepted by the OAuth server.
* SetToken - this takes in the access\_token as a parameter and stores it in the browser's cookies.
* ReadToken - this method is [widely used](https://github.com/ordercloud-api/angular-sdk/blob/master/src/MakeApiCall.js#L6) by the SDK to append an Authentication header to each request
* RemoveToken - this essentially "logs out" the authenticated user by removing the token from cookies.

## **BuyerID Service**

You may have noticed when taking a look at the default [Login Component](https://github.com/ordercloud-api/angular-seed/blob/master/src/app/login/login.js#L111) that we also used something called OrderCloud.BuyerID. This sub-service has two simple methods for setting and getting the active buyer.

Methods:

* Set - this takes in a Buyer Organization ID and stores it using $cookies.
* Get - this method is [widely used](https://github.com/ordercloud-api/angular-sdk/blob/master/src/Generated.js#L49) by the SDK to automatically retrieve the buyerid of the current buyer and set it in the route parameter that is present on many API calls.

## **CatalogID Service**

This is very similar to the BuyerID service. It is used to get and set the catalog ID for the catalog that your buyer organization will be interacting with.

## **Impersonation**

The OrderCloud.io API supports making an API call on behalf of (impersonating) any buyer user within the same organization. There are two different ways to impersonate another user, each of which require the developer to use the .As() method before each sub-service in OrderCloud.io like so:

OrderCloud.As().Me.Get()

The As() [method](https://github.com/ordercloud-api/angular-sdk/blob/master/src/As.js) can work in two different ways:

Developers can generate the token on their own by using

OrderCloud.Users.GetAccessToken()

with the impersonation object.

var impersonation = {  
 "ClientID": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx",  
 "Claims": ["FullAccess"] // (whatever claims you need, set in this array)  
}

This will return with an impersonation token that can be used as an argument directly within the As() chain.

OrderCloud.Users.GetAccessToken(userID, impersonation)  
 .then(function(data){  
 OrderCloud.As(data['access\_token']).Me.ListProducts();  
 });

The second option is storing the returned token manually using the Auth service. The token will then be used by the normal As() chain when nothing is passed as a parameter.

OrderCloud.Users.GetAccessToken(userID, impersonation)  
 .then(function(data) {  
 OrderCloud.Auth.SetImpersonationToken(data['access\_token']);  
 OrderCloud.As().Me.ListProducts();  
 });

In all cases, the impersonation token duration is based off of the API Client AuthTokenDuration and has a max limit of 10 hours.

## **Conclusion**

If you can master using OrderCloud.Auth, OrderCloud.BuyerID, and OrderCloud.As() understanding the rest of the OrderCloud.io Angular SDK is just a matter of internalizing the [API Documentation](http://localhost:3000/docs/api). We have real world examples of virtually every OrderCloud.io sub-service being used in the OrderCloud.io Angular [Components](https://github.com/ordercloud-api/angular-components)